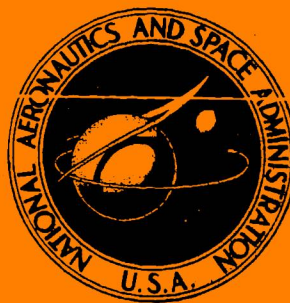


**NASA TECHNICAL  
TRANSLATION**



**NASA TT F-349**

**NASA TT F-349**

# **FUNDAMENTALS OF COMPUTER TECHNOLOGY**

*by Ye. A. Drozdov, V. I. Prokhorov, and A. P. Pyatibratov*

Voyenizdat, Moscow, 1964

**NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • AUGUST 1965**

## FUNDAMENTALS OF COMPUTER TECHNOLOGY

By Ye. A. Drozdov, V. I. Prokhorov, and A. P. Pyatibratov

Translation of "Osnovy vychislitel'noy tekhniki."  
Voyenizdat, Moscow, 1964.

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

---

For sale by the Clearinghouse for Federal Scientific and Technical Information  
Springfield, Virginia 22151 - Price \$5.00

The hardware and operating principles of general-purpose and special-purpose digital computers are described in great detail, with emphasis on large USSR computers of the Ural, Strela, Minsk, Dnpr, and BESM type. All stages of programming, writing, reading, converting, coding, storing, and controlling are explained, including the underlying physical, electronic, and mathematical principles. Numerous wiring diagrams, timing charts, block diagrams, hysteresis loops, and programming samples are given, and merit comparisons of different types of functional blocks are made.

(The translator)

This book presents the design principles of electronic computers and discusses the fundamentals of preparing problems for solution on computers (elements of programming). The material on elements, assemblies, and individual units of the computers has been considerably updated over the first edition, and the terminology and symbols have also been revised.

The book is intended for officers studying digital computers. It may also be helpful to engineers and technologists whose work involves digital computers.

(The author)

---

\* Numbers in the margin indicate pagination in the original foreign text.

This book presents the principles of design and operation of electronic digital computers and of the preparation of problems for solution on such machines.

Chapter I is devoted to the arithmetic and logic principles of the computer, Chapters II-VII directly to computer design, and Chapter VIII to the preparation of problems for computer solution.

The book has been written by a collective of authors, consisting of Candidates in Technical Sciences, Instructors Ye.A.Drozdoz (Chapters V, VI, VII), V.I.Prokhorov (Introduction, Chapters I, II, and VIII) and A.P.Pyatibratov (Chapters III and IV).

From Soviet and foreign sources we have attempted to review, as fully as possible, some electronic digital computers, with special reference to the construction of their units and building blocks from modern miniature elements: semiconductor diodes and triodes, and ferrite cores.

The theoretical level of presentation has to some extent been deliberately lowered to bring it within the grasp of the broad class of readers with education of only high-school level.

The object of this book has been to assist military officers in their study of digital computer technique. It may also be useful to students at Academies and higher schools of military engineering, specializing in digital computer technology.



At the present-day level of development of science and technology, many types of computation are required for solving the most varied problems. The amount and complexity of such computation work is continuously increasing. Computation is particularly difficult in such fields as atomic physics, astronautics, rocket engineering, automation, aircraft construction, etc. Today, the time required to solve a scientific problem depends in most cases on the speed and accuracy of the computations involved. The mathematical apparatus required for the solution of many problems, however, is so complex that a human being is physically unable to solve the problem with the required speed and accuracy, and often cannot solve it at all. Under these conditions, computer technology becomes a powerful tool for rapid technological processes.

Electronic computers are the most rapid means of computation. They perform calculations thousands and tens of thousands of times as fast as the most highly qualified human computers working on electric desk calculators. In several minutes or hours, electronic computers will solve mathematical problems so complex that their ordinary solution would take years.

Before the appearance of electronic computers, computation played only a small part in the solution of problems of scientific research and development and consisted merely in simplified calculations giving approximate results. The problem was for the most part solved by time-consuming and costly experiments.

With the advent of electronic computers, computation has become the method of choice in solving problems in research and development, since computers can perform the most complex calculations and take account of the maximum number of factors characterizing a given subject. This results in substantial financial savings, since a computer can replace a large number of human computers and, what is particularly important, it can achieve a colossal saving of time.

Of course, if a computer can operate so fast that it can replace 10,000 <sup>/6</sup> workers, this does not mean that 10,000 workers can replace a computer. For only a single worker can be used to solve a single mathematical problem, since in mathematical calculations all stages of the computations are performed in succession, one after the other, and each successive stage can be approached only after the completion of the preceding stage. This means that if a digital computer solves some problem in ten hours, it would take the worker, to solve the same problem, 10,000 times as long, i.e., over ten years.

Consequently, the use of electronic computers permits considerable shortening of the time required for computation work, a saving which cannot be achieved by any other means.

Electronic computers are particularly important in cases where the problem can be solved only if the computation can be done at an extremely rapid rate.

Assume that we are to determine the trajectory of a guided space rocket. For this purpose we must calculate the points of the trajectory lying far ahead in the direction of motion of the rocket; only in that case can we estimate the deviation of the rocket from the prescribed direction and apply the necessary midcourse corrections. Such a calculation can be made only by an electronic computer, since workers would require tens of days or several months to calculate a single trajectory, while a rocket takes only three days to reach the moon. The computer will calculate the trajectory in minutes or tens of minutes.

Weather forecasting is another example. To make a forecast, an immense number of calculations must be made in solving the equations of motion of great masses of air. An electronic computer can give an accurate forecast for a day within 1 - 2 hours, while human meteorologists would require several days for the same work.

Computers play an important role in the automation of industrial process control. In automatic control systems the computer is the element that receives information on the course of the controlled process, determines the optimum parameters for all elements of the automatic system, and sends out commands to the devices that establish the required operating conditions.

Like all computers, electronic computers are divided into two classes:

continuous-action machines (simulating machines, analog computers);  
discrete-action machines (digital computers).

In analog computers, the mathematical quantities involved in the solution of a problem are represented by the values of physical quantities: lengths, angles of rotation, voltages, currents, etc. The results of the computations are likewise delivered in the form of physical quantities. The problems are solved by the interaction of moving parts or electrical signals. One of the features of these machines is that each unit of the machine represents one of the quantities involved in the solution of the problem, or solves one definite mathematical relation. For this reason there must be as many such units as there are quantities, and mathematical operations on these variables, in the given problem. Thus, the design of an analog computer is determined by the type of problems to be solved by it. This type of computer, consequently, is always more or less specialized. Examples of such specialized computers in the military field are antiaircraft fire-control stations (PUAZO) or shipboard fire-control stations (PUS). /7

Another feature of analog computers is that the accuracy of the solution is limited. This is explained by the fact that the physical quantities involved in the solution of a problem can be represented only with limited accuracy which, in turn, is determined by the greatest possible manufacturing accuracy of the individual elements and units of the computer.

The lack of versatility of analog computers (each computer can only solve problems of a certain definite class) and the limited accuracy of the calculations are disadvantages of such computers.

A favorable aspect is that the time they take to solve a problem is very short and is determined only by the duration of the transient processes taking place in the machine. Thus, an electronic analog computer solves a problem in hundredths or thousandths of a second, i.e., instantaneously for all intents and purposes. In view of this fact, an analog computer can operate on the real-time scale, permitting its use as a control computer in an automatic control system.

Digital computers operate on quantities that can be represented in the form of discrete, i.e., discontinuous variables. These quantities are represented by digits, and therefore this type of computer is usually called a digital computer. The results of the computations are likewise represented by digits.

The digits in these computers are represented by the aid of elements which can assume a series of sharply limited stable fixed states. Each state of such an element corresponds to a strictly defined digit. The numbers are set up by a set of elements.

The calculations themselves consist in the successive performance of arithmetic operations on the numbers corresponding to the quantities involved in the solution of the problem. The numerical methods developed up to now permit the solution of any mathematical problems to be reduced to the performance of four 8 arithmetic operations. Digital computers can therefore solve practically any mathematical problem.

Thus, the digital machines are distinguished by universality. Any required accuracy of calculation can be obtained with these computers. To increase the accuracy, the number of places in the numbers represented by the computer must be increased, which can be accomplished by increasing the number of elements used to represent the digits. The parts and units of such computers need not be manufactured to high accuracy. The accuracy must only be sufficient to have the elements reliably fixed in the required stable state and be correctly switched from one stable state to another.

The simplest digital machine is the ordinary office abacus. Obviously its accuracy of computation does not depend on the tolerance to which the beads are machined, but on the number of places in the numbers that can be represented on its columns. In exactly the same way, in digital computers the accuracy of calculation is determined by the number of elements used to represent the numbers.

Thus, the fundamental advantages of digital computers over analog computers are versatility and high computational accuracy.

These advantages have been the cause of the sensational development of digital computer technology.

Mechanical, electromechanical and electronic digital computers are in current use.

Machines of the first two types (manual and electrical desk calculators, etc.) have a low operating speed, owing to the low speed of the mechanical parts and their insufficient automation. The entire computational process on such machines is directly controlled by the operator.

Automatic electronic digital computers are free from these disadvantages. Their speed is exceptionally great - thousands and tens of thousands of operations per second. The entire process of computation is automatic, without human intervention, permitting the ultrarapid solution of extremely complex problems.

These advantages have made electronic digital computers the basic computational tool today.

Analog computers are also being developed and improved. They are widely used as control elements for automatic control systems.

As already noted, an analog computer solves a strictly defined class of problems, since each unit can solve only a single definite mathematical relation. Further complication of the problem requires increasing the number of such units, which unavoidably leads to complication of the entire computer. There are also complex problems whose solution on analog computers is disadvantageous and sometimes even impossible. Digital computers are used in such cases. /9

Electronic digital computers are used when high accuracy is required. The nature of the problem is not of substantial significance for such computers, since in principle they can solve any mathematical relation. This feature of the electronic digital computer has been responsible for its wide application in all fields of science and industry.

Electronic digital computers solve the most complex problems in nuclear physics, rocket engineering, astronomy, crystallography, etc. They can also solve complex logical problems of decision theory, data collection and processing, determination of optimum operating conditions, interpretation of texts, etc. Thanks to the ability of digital machines to solve mathematical and logical problems, they are convenient for use as control machines. Such machines are widely used in industry for the automatic control of production processes, and in the military field.

Before the appearance of electronic digital computers, a multitude of timely and urgent problems were not even posed because it would have been impossible to solve them by conventional methods within an acceptable time. The advent of electronic digital computers has opened immense prospects for the solution of such problems.

The appearance of automatic digital computers has led to the rapid development of a new field of mathematics concerned with questions of programming problems for a given class of computers. Programming is a very complex and laborious operation, which requires not only a profound knowledge of the numerical methods of mathematics but also extensive practical experience. The qual-

ity of the programming largely determines the efficiency of computer utilization. In this book, therefore, together with an exposition of the design principles of modern digital computers, we will discuss the problem-programming principles for such computers.

Thanks to the unselfish labor of Soviet scientists and designers, electronic computer engineering is becoming more deeply rooted in our national economy with each passing year. The work of our scientists and engineers has contributed greatly to the development of Soviet computer technology.

The work done by Academician S.A. Lebedev and his colleagues, Yu. Ya. Bazilevskiy, Hero of Socialist Labor, I. S. Bruk, corresponding member USSR Academy of Sciences, Professor L. I. Gutenmakher, engineer B. I. Rameyev, and others, has been of the greatest importance in the development of computer technology in the USSR.

The general-purpose computer BESM-1 (high-speed electronic calculating machine) was developed under the supervision of S.A. Lebedev at the Institute /10 of Precision Mechanics and Computer Technology, USSR Academy of Sciences. On the basis of the BESM-1, the modified computer BESM-2 has been developed and put into series production.

The general-purpose computer "Strela", now successfully used in a number of scientific institutions of the USSR, has been developed under the supervision of Yu. Ya. Bazilevskiy.

In addition to the large general-purpose machines of the type of BESM and "Strela", numerous small and medium general-purpose computers have been developed and put into series production: "Ural-1", "Ural-2", "Minsk-1", "Minsk-2", etc.

The mathematicians A. A. Lyapunov, M. G. Shur-Bure, V. M. Kurochkin and others have rendered great service in developing the mathematical principles of utilization of digital computers and the technique of programming.

The significance of electronic digital computers for the Communist build-up of the USSR would be difficult to overestimate.

The historic decisions of the XXII Congress of the USSR Communist Party, which formulated the magnificent problems of Communist construction in the USSR, provide for the extensive development of mathematical machine building.

The widespread use of digital computers will encourage still greater development in Soviet science and technology and will help to strengthen the defensive power of the USSR.

# ARITHMETIC AND LOGICAL PRINCIPLES OF ELECTRONIC DIGITAL COMPUTERS

## Section 1. Positional Systems of Notation

The term "system of notation" is usually taken to mean a method of writing numbers by means of digital symbols (digits). We distinguish positional and nonpositional systems of notation. The decimal and Roman systems are widely used today. The former is positional, the latter nonpositional.

A system of notation is called positional if the value of each digit used to write a given number depends on its position in the series of digits representing that number. For example, in the number 777, the first 7 on the left indicates the number of hundreds contained in the number, the second the number of tens, and the third the number of units.

A system of notation in which the value of a digit does not depend on its position in the series of digits representing a number is called nonpositional. Thus, in the number XXX, the digit X in any place indicates the number ten.

Digital computers use only positional systems of notation, since nonpositional systems are inconvenient for calculation.

The decimal system of notation. The decimal system of notation is generally used today. It is so called because, to write a number in it, we use ten different digits: 0, 1, 2, 3, 4, 5, 6, 7, 8 and 9, representing the integers from zero to nine. The number ten, which is the base of the system (the number of digits used in a system is called the base of the system) is represented by two digits as 10. Every other number is written in the form of a sequence of digits, divided by a point into an integral part and a fractional part.

Take, for example, the number two thousand one hundred forty-five and six tenths. In the decimal notation familiar to us, this number is represented by the following sequence of digits: 2145.6.

Let us analyze the meaning of this number:

$$2145.6 = 2 \cdot 10^3 + 1 \cdot 10^2 + 4 \cdot 10^1 + 5 \cdot 10^0 + 6 \cdot 10^{-1}.$$

$\downarrow \quad \quad \downarrow \quad \quad \downarrow \quad \quad \downarrow \quad \quad \downarrow$   
 2            1            4            5            6

Thus, a decimal number is represented by a sum of various powers of ten 12 with the corresponding coefficients (indicated by arrows). These coefficients form the number 2145.6 in form of an abbreviated notation.

There are other positional systems of notation besides the decimal. By

taking various numbers as the base of the system - two, three, five, eight, or some other number - we can obtain respectively the binary, ternary, quinary, octal and other systems. In any of these systems, as in the decimal system, a number will be a sum of powers of the base of the system with the corresponding coefficients:

$$N_q = K_n q^n + K_{n-1} q^{n-1} + \dots + K_2 q^2 + K_1 q^1 + K_0 q^0 + K_{-1} q^{-1} + \dots \quad (1)$$

where

$N_q$  is a number in the  $q$ -ary system of notation;  
 $q$  is the base of the system;  
 $n$  is the number of places.

The abbreviated way of writing the number  $N_q$  will be of the form

$$N_q = K_n K_{n-1} \dots K_1 K_0 K_{-1} \dots$$

In antiquity, systems of notation to the base 5, 10, 12, 60, etc. were used.

The octal system of notation. In the octal system, eight digits: 0, 1, 2, 3, 4, 5, 6, and 7, are used to write numbers. They designate the integers from zero to seven. The number eight (the base of the system) is written with two digits in the form 10.

Let us write the number sixty-nine in the octal system. According to eq.(1), this number must be expanded in powers of eight since the base of the system is eight.

$$\begin{array}{rcc} \text{Sixty-nine} & = & 1 \cdot 8^2 + 0 \cdot 8^1 + 5 \cdot 8^0 \\ & & \downarrow \quad \downarrow \quad \downarrow \\ & & 1 \quad 0 \quad 5 \end{array}$$

Thus, in the octal system the abbreviated notation for the number sixty-nine will read 105.

Let us, whenever necessary, use a subscript to indicate the base of the system in which the expression is written. Then we may write

$$69_{(10)} = 105_{(8)}$$

The binary system of notation. The smallest number of digits that can be used in a system of notation is two.

The binary system has only two digits, 0 and 1. The base of this system, two, is written out as 10, and any other number in the form of a combination of zeros and ones.

In accordance with eq.(1), the binary number  $N_2$  is the sum:

/13

$$N_2 = K_n \cdot 2^n + K_{n-1} \cdot 2^{n-1} + \dots + K_1 \cdot 2^1 + K_0 \cdot 2^0 + K_{-1} \cdot 2^{-1} + \dots$$

Here, the coefficients K can take only two values: 0 and 1.

Let us write the number  $69_{(10)}$  in the binary system of notation:

$$69_{(10)} = 1 \cdot 2^6 + 0 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$

↓  
1

↓  
0

↓  
0

↓  
0

↓  
1

↓  
0

↓  
1

Hence,

$$69_{(10)} = 1000101_{(2)}$$

A distinctive feature of the writing of numbers in the binary system is that the notation for a number is long, but contains few different digits (only two: 0 and 1).

TABLE 1  
EXPRESSION OF NUMBERS IN VARIOUS POSITIONAL  
SYSTEMS OF NOTATION

System of Notation				
Decimal	Octal	Quin-ary	Ternary	Binary
0	0	0	0	0
1	1	1	1	1
2	2	2	2	10
3	3	3	10	11
4	4	4	11	100
5	5	10	12	101
6	6	11	20	110
7	7	12	21	111
8	10	13	22	1000
9	11	14	100	1001
10	12	20	101	1010

It is easy to explain how numbers will be written in systems with other bases.

Let us see, for example, how the number one hundred seventy-eight will look in the quinary system of notation. We expand in powers of five:

$$178_{(10)} = 1 \cdot 5^3 + 2 \cdot 5^2 + 0 \cdot 5^1 + 3 \cdot 5^0$$

Consequently,

$$178_{(10)} = 1203_{(5)}$$



If the base is greater than ten, the available Arabic digits are not enough and other digits must be thought up.

Table 1 gives, for comparison, the notation of the numbers from zero to ten in various systems of notation.

## Section 2. Arithmetic of Binary Numbers

/1

The binary system of notation is remarkable, as compared to other positional systems, in that the arithmetic operations are simplest of all.

To perform the four arithmetical operations in any system of notation, it is necessary to know the addition, subtraction, and multiplication Tables. In the binary system, these Tables are exceptionally simple. Each of them consists of only four lines.

Addition of binary numbers. The addition Table for binary numbers is as follows (Table 2).

TABLE 2

BINARY ADDITION TABLE

$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 10$

Using this Table, the addition of binary numbers can be performed by the same rule that is used in the addition of decimal numbers.

As an example, find the sum of the two numbers 1010 and 1011. Writing these numbers in columns, we add them by the general rule

$$\begin{array}{r} + 1010 \text{ (ten)} \\ + 1011 \text{ (eleven)} \\ \hline 10101 \text{ (twenty-one).} \end{array}$$

The addition of two binary numbers involves no difficulty. It is only necessary to bear in mind that  $1 + 1$  gives a zero in the given column and yields a carry of one to the next column.

The simultaneous addition of three or more binary numbers is somewhat more complicated. In this case one must carefully follow the ones carried over to

higher columns during addition, since these ones can go not only to the next higher columns but also beyond.

Let us add the following four binary numbers: 1010, 1111, 1100, 0111. The carry-overs formed an addition into the corresponding columns will be indicated by arrows. /15

$$\begin{array}{r}
 \begin{array}{c} \downarrow \leftarrow | \leftarrow | \\ 1010 \text{ (ten)} \\ 1111 \text{ (fifteen)} \\ 1100 \text{ (twenty)} \\ 0111 \text{ (seven)} \end{array} \\
 + \\
 \hline
 101100 \text{ (forty-four)}
 \end{array}$$

The sum of the second places of the summands, counting the carry one, is four, i.e., it is the three-place binary number 100. Consequently, the second place of the sum will again contain a 0, but the carry one must jump a place, as shown by the arrow. The sum of the higher places of the summands, including the two carry ones, is five (101).

Subtraction of binary numbers. We present the subtraction Table for binary numbers (Table 3).

TABLE 3

BINARY SUBTRACTION TABLE

0-0=0
1-0=1
1-1=0
10-1=1

Binary numbers are subtracted by the usual rule, writing the numbers in columns. Let us find the difference of the numbers 10101 and 1010.

$$\begin{array}{r}
 10101 \text{ (twenty-one)} \\
 - 1010 \text{ (ten)} \\
 \hline
 1011 \text{ (eleven)}
 \end{array}$$

In subtraction, it must be remembered that a one borrowed from the next higher place gives two ones in the lower place. If there are zeros in the adjacent higher places, then one must be borrowed after jumping several places.

In this case, a one borrowed from the next higher significant place gives two ones in the lower place and ones in all the zero places between the lower place and the higher place from which the one was borrowed, for example:

$$\begin{array}{r}
 \text{Minuend} \quad 0111 + 1 \leftarrow \text{After borrowing one} \\
 \quad \quad \quad 1000 \quad \quad \quad \text{(eight)} \\
 \text{Subtrahend} \quad 11 \quad \quad \quad \text{(three)} \\
 \hline
 \text{Difference} \quad 101 \quad \quad \quad \text{(five)}
 \end{array}$$

Multiplication of binary numbers. We present the multiplication Table /16 of binary numbers (Table 4).

TABLE 4  
BINARY MULTIPLICATION TABLE

0·0 = 0
0·1 = 0
1·0 = 0
1·1 = 1

Binary numbers are multiplied by the same rules as decimal numbers. In multiplying, we use multiplication and addition Tables.

$$\begin{array}{r}
 \times \quad 10111.01 \left(23\frac{1}{4}\right) \\
 \quad \quad 10.11 \left(2\frac{3}{4}\right) \\
 \hline
 \quad \quad 1011101 \\
 + \quad 1011101 \\
 \hline
 \quad 1011101 \\
 \hline
 11111.1111 \left(63\frac{15}{16}\right).
 \end{array}$$

Division of binary numbers. In the division of binary numbers we use binary multiplication and subtraction Tables.

$$\begin{array}{r}
 - 110101110 (430) \\
 \quad 1010 \\
 \hline
 \quad \quad 1101 \\
 \quad \quad - 1010 \\
 \quad \quad \hline
 \quad \quad \quad 1111 \\
 \quad \quad \quad - 1010 \\
 \quad \quad \quad \hline
 \quad \quad \quad \quad 1010 \\
 \quad \quad \quad \quad - 1010 \\
 \quad \quad \quad \quad \hline
 \quad \quad \quad \quad \quad 0000
 \end{array}
 \qquad
 \begin{array}{r}
 | \quad 1010 (10) \\
 101011 (43).
 \end{array}$$

It will be clear from the above examples that the rules for performing the arithmetic operations are the same in the decimal and binary systems. In the binary system, however, the arithmetic operations are far simpler. Multiplication and division are particularly simple.

### Section 3. The Systems of Notation Used in Digital Computers

To represent numbers in digital computers, the elements used are capable of being in one of several fixed and sharply defined states. The number of these states must equal the number of digits in the system of notation used in the machine; to each digit there corresponds a strictly determined state of the element.

In an electronic digital computer, the elements used to represent digits are electron tubes, capacitors, relays, ferromagnetic surfaces, etc. As a rule these can be only in one of two distinctly manifest stable states. An electron tube may conduct current (tube open) or not conduct it (tube blocked), a capacitor may be charged or discharged, a relay on or off, a ferromagnetic surface magnetized or demagnetized, etc.

Thus, the elements of an electronic digital computer, by their physical nature, may be in only one of two states. Such elements are usually called flip-flop or bipoistional.

We are most accustomed to the decimal system. To use this system, however, in electronic digital computers, it would be necessary to devise elements with ten stable positions to represent the digits. Such elements in themselves would be rather complex circuits, built up from flip-flop elements. The machine would then be highly complex. The decimal system is therefore inconvenient for use in electronic digital computers.

The binary system. Flip-flop elements operate on the simplest and most reliable principle of action - "yes" or "no" (on or off). By their aid it is very easy to represent the places of a binary number. One of the stable states of the element is taken as the representation of the digit 0, and the other as the representation of the digit 1.

In using the binary system of notation, the elements representing the digits come out simplest of all.

The binary system, as we have already established, also possesses immense advantages in arithmetic. This permits considerable simplification of the design of the arithmetic and memory units.

Another great advantage of the binary system is that it permits the use of the apparatus of mathematical logic (see Section 11) in the analysis and synthesis of the functional circuits of the computer and in solving various logical problems.

In view of these advantages, the binary system today is the principal

11

system of notation used in electronic digital computers.

The use of the binary system, however, also involves certain inconveniences.

Since the decimal system is universally used, all initial data and computational results are written out in this system. These initial data must therefore first be translated from the decimal system into the binary system, and after the computer has solved the problem, the result must again be translated from the binary system into the decimal. This complicates the operation of digital computers. However, in solving most mathematical and logical problems, innumerable operations must be performed on the numbers (thousands, tens of thousands, and even hundreds of thousands of operations), while the amount of input data and results is relatively small (tens, hundreds, and seldom thousands), so that this disadvantage of digital computers cannot be a major obstacle to their use.

The octal system of notation. The octal system of notation is also widely used in digital computers. It is employed as an auxiliary system in preparing a problem for solution (programming). This system is convenient because the octal notation for any number is only a third as long as its binary notation, and the conversion of the numbers from one system into the other is simple and can be performed by a purely mechanical method (see Section 4).

The binary-coded decimal system of notation. Besides the binary and octal systems, the so-called binary-coded decimal system is also used in digital computers. Like the octal system, it plays an auxiliary role.

In this system, numbers are written as follows: The decimal number is taken as the base. Then each digit of the decimal number (from 0 to 9) is written out in the form of its binary equivalent. It will be seen from Table 1 that not more than four binary places will be required for such notation. The four-place binary number representing a decimal digit is called a tetrad.

To write a given decimal number in the binary-coded decimal system, each of its digits must be replaced by the tetrad corresponding to it. Let us take for example the decimal number 2467.39 and write it in the binary-coded decimal system.

Let us rewrite this decimal number with more space between the digits, and under each digit let us place the tetrad corresponding to it:

2	4	6	7	,	3	9
0010	0100	0110	0111	.	0011	1001

Thus the decimal number 2467.39 equals the binary-coded decimal number 0010010001100111. 00111001.

The conversion of numbers from the decimal system into the binary-coded decimal system does not require calculation. The decimal digits may therefore

be coded in advance into their binary equivalents, and the numbers may then be mechanically translated from the decimal system into the binary-coded decimal system.

The inverse conversion of a number from the binary-coded decimal system into the decimal system is also very simple. The binary-coded decimal system must first be broken down, to the left and right of the point, into tetrads or groups of four digits each, and then each tetrad must be replaced by the corresponding decimal digit.

For example, given the binary-coded decimal number:

100101010100011010000110. 010001110010.

We break this up into tetrads and replace each tetrad by the decimal digit:

1001 0101 0100 0110 1000 0110. 0100 0111 0010 = 954686.472

In electronic digital computers, the binary-coded decimal system is used as an intermediate system between the decimal and binary systems of notation. Before introducing the original data into the computer, the decimal digits are converted by special devices into binary-coded decimal digits. Then, by a special program, the computer itself translates the binary-coded decimal digits into binary digits. After completing the calculations, the computer automatically, by a special program, translates its result from the binary into the binary-coded decimal notation, after which special devices give the final result in decimal notation.

There are some computers in which the calculations themselves are performed in the binary-coded decimal system.

#### Section 4. Conversion of Numbers from the One Positional System of Notation into Another

The tabular method. This method consists essentially in representing a number by (eq.(1) in the form of a sum of powers of the base of the system in which the number is to be written. Then, by the aid of the coefficients of the powers of the base, the number is written out in abbreviated form. In expanding the number, Tables of powers of the base are used. Hence the name "tabular method".

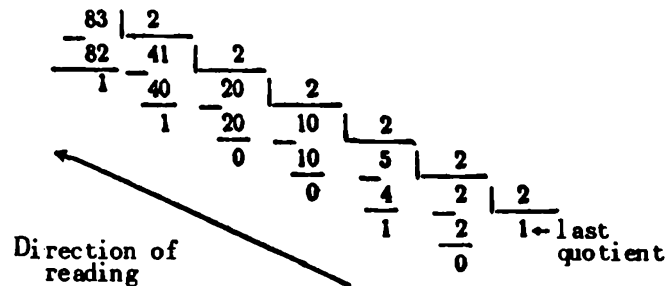
The tabular method of conversion does not require complicated calculations, and herein lies its advantage.

General rule for conversion of integers. To translate an integer from one positional system into another, it must be successively divided by the base  $q$  of the system into which it is being translated, until a quotient less than  $q$  is obtained. The number in the new system is written in the form of the remainders, starting with the last one. The last quotient gives the most significant digit of the number.

We shall illustrate this rule by an example.

Example. Convert the decimal number 83 into the binary system

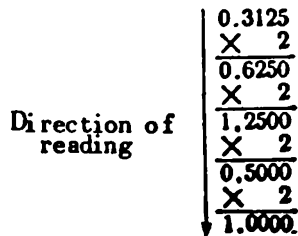
/20



Thus,  $83_{(10)} = 1010011_{(2)}$ .

General rule for converting proper fractions. To translate a proper fraction from one positional system into another, it must be successively multiplied by the base of the system into which it is being translated. Only the fractional parts are multiplied. The fraction in the new system is written in the form of the integral parts of the products so obtained, beginning with the first one.

Example 1. Convert the decimal 0.3125 into a binary



Consequently,  $0.3125_{(10)} = 0.0101_{(2)}$ .

Example 2. Convert the decimal 0.12 into a binary

$$\begin{array}{r}
 0.12 \\
 \times 2 \\
 \hline
 0.24 \\
 \times 2 \\
 \hline
 0.48 \\
 \times 2 \\
 \hline
 0.96 \\
 \times 2 \\
 \hline
 1.92 \\
 \times 2 \\
 \hline
 1.84 \\
 \times 2 \\
 \hline
 1.68 \\
 \times 2 \\
 \hline
 1.36 \\
 \times 2 \\
 \hline
 0.72 \text{ etc.}
 \end{array}$$

Thus, the decimal 0.12 equals the binary 0.0001111, rounding off to one hundred twenty-eight.

General rule for converting improper fractions. In translating improper <sup>/21</sup>fractions, the integral and fractional parts are separately translated by the corresponding general rules.

Note. In performing conversions by the general rules, all the required arithmetic operations are performed in the system of notation in which the number being translated had originally been written.

Conversion of numbers from the octal system into the binary, and vice versa. Octal numbers can easily and rapidly be translated into binary and vice versa by special rules which will be given below.

First let us convince ourselves of the validity of the following two lemmas:

- a) every one-valued octal number can be written in the form of a three-digit binary number;
- b) every three-digit binary number can be written in the form of a single-digit octal number.

The validity of these lemmas can easily be verified from Table 1.

The rule for converting numbers from the octal system of notation into the binary system is as follows: to translate octal numbers into the binary system it is sufficient to replace each octal digit by its equivalent three-digit binary number.

Example. Translate the octal number 14035 into the binary system. Each digit of this number is replaced by the three-digit binary number equivalent to it:

1	4	0	3	5
001	100	000	011	101

Thus,  $14035_{(8)} = 1100000011101_{(2)}$ .

The rule for converting numbers from the binary system of notation into the octal system is as follows: to translate binary numbers into the octal system, the binary number must be broken up into groups of three digits, or triads, to the left and right of the point; if the triad on the extreme left or the extreme right is incomplete, a zero is added to it; each triad of binary digits is then replaced by the octal digit equivalent to it.

Example. Convert the binary number 111110101.1011101 into the octal system.

Let us break up the number into triads of digits to the left and right of the point:

11 111 010. 101 110 1.



The last triads on the right and left are incomplete; we complete them by adding zeros:

011 111 010. 101 110 100.

Each triad is now replaced by the equivalent octal digit, thus yielding the octal number 372.564.

It will be clear from these rules that it is a feature of the conversion of octal numbers into binary numbers and inversely that no calculations are required for this purpose. This conversion can be performed in a purely mechanical way.

Let us imagine that we have a typewriter with single-digit octal numbers /22 on the keys and the corresponding three-digit binary numbers on the typebars. If any octal number is selected on the keyboard, the machine will automatically print it in the binary system. Binary numbers can be similarly translated into octal numbers, except that in this case the three-digit binary numbers are marked on the keys, and the corresponding octal numbers are on the typebars.

The octal system is used in programming to write in the numbers of the commands or instructions, the addresses of the numbers, the codes of operations, the initial data for a solution, and several other constants. This entry is convenient because it occupies fewer places than the binary notation. During the input of the information to the computer by means of a keyboard device, the octal notation is automatically converted into binary.

Conversion from the decimal system of notation into the binary. If decimal numbers are to be converted into binary, the following is the procedure used in practice: the decimal number is first translated into an octal number according to general rules after which the octal number is converted into a binary number. This is done because decimal numbers are more rapidly converted into octal numbers than directly into binary numbers, and the transition from octal numbers to binary equivalents is very simple.

Conversion into the decimal system of notation. On passage from any system of notation to the decimal system, a number is usually represented in the form of a sum of powers of the base, using positional writing. The value of the sum is then calculated. This procedure is adopted because we are more familiar with decimal notation and have no trouble in performing calculations in this system.

## Section 5. Representation of Numbers on Digital Computers

In digital computers, depending on their design, two forms of number representation are used:

- a) natural, or fixed-point;
- b) normal, or floating-point.

Accordingly, computers are classed as fixed-point or floating-point.

Natural form. In the natural form of numbers representation, the position of the point, which separates the integral part of the number from the fractional part, is constant for all numbers with which the computer operates. In designing a computer, the number of places given to the integral part and the number of places to the fractional part is established in advance.

Assume that a digital computer is designed to represent a six-place decimal number, with three places assigned to the integral part of the number and three places to the fractional part or, expressed differently, with the point fixed after the third digit. In this case, numbers in the following range /23 can be represented in the computer:

$$\begin{array}{r} +999.999 \\ \cdot \cdot \cdot \\ +000.001 \\ 000.000 \\ -000.001 \\ \cdot \cdot \cdot \\ -999.999 \end{array}$$

An analysis of these numbers readily shows the drawback of computers of the fixed-point type.

1) The range of numbers representation is relatively small (in our example from -999.999 to +999.999). Any number having an absolute value less than the minimum that the computer is able to represent is written in the form of a zero (the so-called machine's idea of zero).

2) The number obtained as the result of the calculation must not exceed in absolute value the maximum number that can be represented by the computer (in our example 999.999); otherwise the most significant digits of the number will be lost, and the computational results will be distorted. This phenomenon is known as overflow.

To prevent overflow, the initial data must first be multiplied by the corresponding scale factors.

The selection of scale factors is quite intricate and depends largely on the experience of the mathematician who prepares the problem for solution.

The point is usually fixed before the first (most significant) figure; for this purpose, all the quantities involved in the problem must be less than unity. Fixing the point in this way somewhat facilitates selection of the scale factor, since in multiplication which occurs frequently overflow is prevented, because of the fact that the product in this case will not be greater in value than any of the co-factors.

Normal form. In the normal form of numbers representation, any number is represented in the form of two groups of digits. The first group is known as the mantissa, and the second group as the exponent.

To illustrate the representation of a number in the normal form, let us take a specific example. Assume that we have the decimal number 724.863. This number can be represented as the product of two co-factors:

$$724.863 = 72.4863 \cdot 10^1,$$

or

$$724.863 = 0.0724863 \cdot 10^4.$$

Obviously, in the general case any number  $N$  can be written in the form /24 of a product of two co-factors:

$$N = mq^p, \quad (2)$$

where  $m$  is a fractional number;

$q$  is the base of the system of notation;

$p$  is an integer.

The number  $N$  is considered to be represented in the normal form if the first co-factor  $m$  is a proper fraction, i.e., if the condition  $|m| < 1$  is satisfied. In this case, the co-factor  $m$  is the mantissa and the superscript  $p$  the exponent of the number.

To represent it in the normal form, we must transform the number 724.863 as follows:

$$724.863 = 0.724863 \cdot 10^3.$$

Assume that six places are assigned in a digital computer for the representation of the mantissa and two places for the representation of the exponent. Then, in the normal form the number 724.863 will be written as

$$+ 724863 + 03$$

It should be noted that the representation of the number in the normal form is non-unique; the following forms are also correct:

$$724.863 = 0.0724863 \cdot 10^4,$$

$$724.863 = 0.00724863 \cdot 10^5 \text{ etc.}$$

The exponent of a number need not be positive and may also be negative. Thus, the number 0.000342527 can be represented in the form of the product

$$0.000342527 = 0.342527 \cdot 10^{-3}.$$

Accordingly, the number in the normal form will be written as

$$+ 342527 - 03.$$

Thus, the exponent of a number written in the normal form shows the position of the point if the number is represented by the set of digits of the mantissa. Since, with different exponents, the position of the point will be different, computers using the normal form of numbers representation are known as floating-point computers.

Up to now all our arguments have been proved by examples in the decimal system of notation. Obviously, all above statements are fully applicable to the representation of numbers in any system of notation. For example, the binary number 1101.01 can be represented by eq.(2) as follows:

$$1101.01 = 0.110101 \cdot 10^{100}.$$

Here 10 is the base of the system, two.

In the normal form, the number is written as

/25

$$+ 110101 + 100.$$

We distinguish standardized and nonstandardized normal numbers. If a non-zero digit occupies the first place of the mantissa, the number is called standardized, but if this first digit is a zero, the number is called non-standardized.

It is preferable to store numbers in the normalized form in a computer, since the last places of the mantissa will not be lost in that case.

The advantage of floating-point computers lies in the fact that they offer a rather broad range of numbers representation, without the use of scale factors. Their disadvantages, by comparison with fixed-point machines, consist in that more elements are required to represent the numbers since, besides the elements required to represent the mantissa, additional elements are needed to represent the exponent.

No one form of representing numbers is invariably preferable. In large general-purpose digital computers designed to solve a broad range of problems, the normal form of notation is generally used. This makes for greater capacity and a higher computational accuracy. The additional complexity of the hardware and the greater dimensions for a given class of computers are of secondary importance.

Special-purpose and small digital machines are used to solve a more narrow range of problems, for which the order of magnitude and the required accuracy of calculation are usually established in advance. Fixed-point representation is therefore used in such computers, thus making them less complicated and consequently less expensive.

In some cases, computers with both types of representation are used. In solving a problem, the form most rational for the specific case is used.

Representing the signs of numbers on computers. In computation, operating with both positive and negative binary numbers is necessary. Since their writing must not differ, special symbols are introduced to represent the signs of the numbers: a positive sign is indicated by a zero, a negative sign by a one.

A special binary location is assigned to represent the signs of the numbers, usually in front of the number proper. In fixed-point computers this location is in the integers, if the point is fixed before the first significant figure. For example, the binary numbers +0.001011 and -0.001011, differing in sign, are written in the natural form of

0001011,  
1001011.

In floating-point machines, two locations are assigned to represent the <sup>26</sup> signs, one for the sign of the mantissa, the other for the sign of the exponent. Assume that six places are assigned in the computer to represent the mantissa and three places to represent the exponent. Then, the binary number +0.001011 =  $0.10110 \times 10^{-10}$  is written as

Mantissa	Exponent
010110	110

## Section 6. Input of Numbers to the Memory Locations of a Digital Computer

The memory location or cell of a computer is a device designed to store a number. The maximum number of digits that can be stored in such a cell or

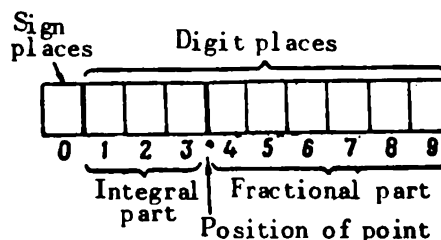


Fig.1 Schematic Diagram of Memory Location of a Computer with Fixed-Point Representation



Fig.2 Schematic Diagram of Input of the Binary Number -110.000101 in the Memory Location of a Computer with Fixed-Point Representation

location depends on the purpose and design of the computer. The locations of general-purpose digital computers have room for about forty binary digits. These locations, in the structure of the computer, may be dispersed among the various parts of the computer.

Input of numbers in the memory locations of a fixed-point computer. A memory location of a fixed-point computer has a sign place and digital places in which the integral and fractional parts of a number are stored.

Figure 1 is a schematic representation of such a location, with three blocks or places for storage of the integral part of a number and six blocks or places for storage of its fractional part. In reality, the locations have far more places, but for clarity we confine ourselves here to nine, which will suffice to illustrate the principle of input.

Consider the following cases of entry from the example of our memory location.

In the zero place of the location (the sign place), the sign of the number is entered. The number itself is written into the digital places of the location, each place of the integral and fractional parts of the number being entered in a strictly determinate position, depending on its distance from the point. Figure 2 is a schematic diagram of input of the binary number /27  
-110.000101 in a memory location.

We noted above that, to facilitate the selection of scale factors, most

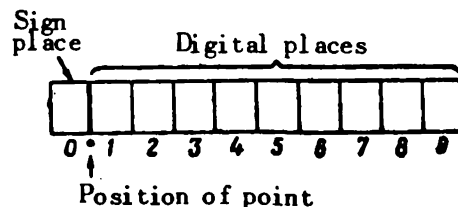


Fig.3 Schematic Diagram of Memory Location of Computer with the Point Fixed in Front of First Digital Place

fixed-point computers have the point fixed before the first digital place of the location (immediately following the sign place). Figure 3 shows a diagram of such a memory location.

Write-in of numbers in a memory location of a floating-point computer. A memory location of a floating-point computer consists of two parts: one for input of the mantissa, the other for input of the exponent. Figure 4 is a

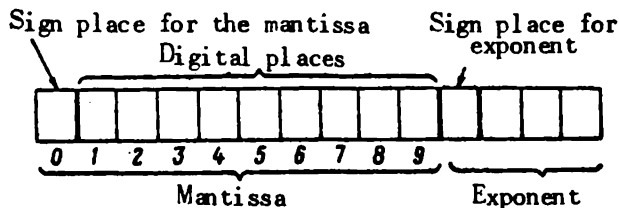


Fig.4 Schematic Diagram of Memory Location of a Floating-Point Computer

schematic diagram of such a memory location; the cell has ten places (including the sign place) for input of the mantissa and four places (together with the sign place) for input of the exponent. In actual computers 30 - 36 places are

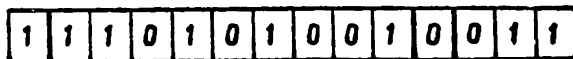


Fig.5 Schematic Diagram of Entry of the Binary Number -110.101001 in Standardized Form in a Memory Location of a Floating-Point Computer

assigned for representation of the mantissa and 6 - 7 places for representation of the exponent.

The signs of the mantissa and of the exponent are entered in the corresponding sign places of the location, the mantissa in the digital places



Fig.6 Schematic Diagram of Entry of the Binary Number -110.101001 in Nonstandardized Form in the Memory Location of a Floating-Point Computer

(Fig.4 shows nine of them), and the absolute value of the exponent in the last places of the location (three in our case).

The mantissa of a number in normal writing is always a proper fraction

and is fed to a location in the same way as a number is entered in the location of a computer with the point fixed before the first digital place. In cases where the number of digits of the mantissa being entered exceeds the number of digital places of the location, the last figures of the mantissa are lost.

Figure 5 gives a schematic diagram for input, in a memory location of a floating-point computer, of the binary number  $-110.101001 = -0.110101001 \times 10^{+11}$ . The number is entered in standardized form. Figure 6 is a schematic diagram of the entry of the same number represented in the form  $-110.101001 = -0.000110101001 \times 10^{+110}$ . The number is represented in nonstandardized writing, and the last three figures of the mantissa are lost. /28

## Section 7. Representation of Negative Numbers

In electronic digital computers the circuits performing the operation of addition are far simpler and more compact than those performing the operation of subtraction. For this reason, digital machines ordinarily use only addition circuits, and the operation of subtraction is replaced by addition of specially selected number codes.

The following number codes are used: direct, inverse or base-minus-ones, and (true) complement. The direct code is used in multiplication and division, while the latter two codes are used to replace the operation of subtraction by addition. There are also modified base-minus-ones and complement codes.

Positive numbers are represented in the same way in all three codes: direct, inverse, and complement. The inverse and the complement codes of negative numbers, as well as their multiplication, differ.

In digital computers, binary numbers that are proper fractions are usually represented by various codes.

Direct code. The number  $X$  in the direct code is denoted by the symbol  $[X]_{dir}$ . Let  $X$  be the proper binary, positive or negative. The direct code of the number  $X$  is obtained by the following rule:

If  $X = +0, X_1X_2 \dots X_1 \dots X_n$ , where  $X_1, X_2, \dots, X_1$  are binary digits, /29 then

$$[X]_{dir} = X = 0, X_1X_2X_3 \dots X_n$$

However, if  $X = -0, X_1X_2X_3 \dots X_n$ , then

$$[X]_{dir} = 1, X_1X_2X_3 \dots X_n$$

Thus, the direct code of a binary number coincides in its representation with the writing of the number itself, but there is either a 0 or a 1 in the sign place, depending on whether the number is positive or negative.

The inverse or base-minus-ones code. The base-minus-ones code of the number  $X$  is denoted by  $[X]_{inv}$ . As already noted, the base-minus-ones code or



inverse code of a positive number is the same as its direct code. Therefore, for  $X > 0$ , we have

$$[X]_{inv} = [X]_{dir} = X.$$

For a negative number the inverse code is obtained by the following rule: in the sign place of the number we write one, and in the digital places the zeros are replaced by ones and the ones by zeros.

Thus, if we have the negative number

$$X = -0, X_1 X_2 \dots X_i \dots X_n$$

its representation in the inverse code will be

$$[X]_{inv} = 1, \bar{X}_1 \bar{X}_2 \dots \bar{X}_i \dots \bar{X}_n$$

where  $\bar{X}_i = 1$  if  $X_i = 0$  and  $\bar{X}_i = 0$  if  $X_i = 1$ .

Example.  $X = -0.100010$ ;  $[X]_{inv} = 1.011101$ .

In digital computers, the addition of inverse codes by the appropriate rules gives the inverse code of the sum.

Complement code. The complement code of the number  $X$  is denoted by  $[X]_{com}$ .

For  $X > 0$ ,

$$[X]_{com} = [X]_{dir} = X.$$

The complement code of a negative number is obtained by the following rule: in the sign place of the number we write a one, while in all the digital places the zeros are replaced by ones, and the ones by zeros, and one is added to the lowest digital place.

Thus, the complement code of the negative number

$$X = -0, X_1 X_2 X_3 \dots X_i \dots X_n \text{ will be}$$

$$[X]_{com} = 1, \bar{X}_1 \bar{X}_2 \dots \bar{X}_i \dots \bar{X}_n + \underbrace{0,000\dots 1}_{n \text{ places}}$$

where

$$\bar{X}_i = 1 \text{ if } X_i = 0, \text{ and } \bar{X}_i = 0 \text{ if } X_i = 1.$$

Example.  $X = -0.110111$ ;  $[X]_{com} = 1.001000 + 0.000001 = 1.001001$ .

/30

It is easy to prove that the complement code of a negative binary fraction is its complement to two, i.e.,

$$[X]_{com} = 10 + X,$$

where 10 denotes two in the binary system.

When complement codes are added in the computer by the appropriate rules, we obtain the complement code of the sum.

Besides the inverse and complement codes, some digital computers use modified inverse and complement codes. As we shall show below, these are more convenient to detect overflow, which may occur on addition.

In modified codes the signs of the numbers are represented by two binary places: a plus is represented by two zeros and a minus by two ones.

The modified inverse code. The modified inverse code of the number  $X$  is denoted by  $[X]_{1,v}^m$ . Proper binary fractions are translated into the modified inverse code by the same rules as into the inverse code. The only difference is that two places are used to represent the sign in the modified inverse code.

Example.  $X = -0.100010$ ;  $[X]_{1,v}^m = 11.011101$ .

Modified complement code. The number  $X$  in the modified complement code is denoted by  $[X]_{0,v}^m$ . The numbers are translated into this code in the same manner as into the complement code, but two places are used to indicate the sign.

Example.  $X = -0.110111$ ;  $[X]_{0,v}^m = 11.001001$ .

## Section 8. Addition of Numbers on Computers

Addition of numbers on fixed-point computers. In fixed-point computers, numbers are added in complement or inverse code. It is compulsory here to observe the condition that each of the summands and their sum in absolute value shall always be less than unity.

Addition of numbers in complement code. The complement codes of the numbers are added, place by place, and the sign places are added like the integer places.

A feature of the addition of complement codes is that a carry one, formed on addition of the sign places, is not taken into account, i.e., is discarded.

Let the numbers  $X$  and  $Y$  to be added be proper fractions and  $|X + Y| < 1$  (absolute value of the sum is less than 1). When the complement codes of numbers satisfying these conditions are added, four different cases may occur. /31  
Let us consider them on the basis of examples.

Case 1.  $X > 0$ ;  $Y > 0$  (summand is positive).

Direct Code	Complement Code	Addition in Compl. Code
$X = 0.101001 \rightarrow$	$[X] = 0.101001 \rightarrow$	$0.101001$
$+ Y = 0.001101 \rightarrow$	$[Y]_{com} = 0.001101 \rightarrow$	$+ 0.001101$
$X + Y = 0.110110 \rightarrow$	$[X + Y]_{com} = 0.110110 \leftarrow$	$0.110110$

In the left column, the numbers are added in the direct code; then both numbers and their sum are converted into the complement code (center column). In the right column, the two numbers are added in the complement code. Comparing the sum obtained with the third line of the center column, we convince ourselves that these numbers are the same. This case shows no peculiarities of addition in the complement code, because the summands are positive and their representations in the complement and direct codes are identical.

Case 2.  $X > 0; Y < 0; X + Y < 0$ .

One of the summands is positive, the other is negative, and their sum is negative.

In exactly the same way as in the last case, we write in the left column the direct codes of the summands and find their sum. We then translate the summands and the sum into the complement code and in the right column perform the addition in the complement code.

Direct Code	Complement Code	Addition in Compl. Code
$X = 0.001101 \rightarrow$	$[X]_{com} = 0.001101 \rightarrow$	$0.001101$
$+ Y = -0.101001 \rightarrow$	$[Y]_{com} = 1.010111 \rightarrow$	$+ 1.010111$
$X + Y = -0.011100 \rightarrow$	$[X + Y]_{com} = 1.100100 \leftarrow$	$1.100100$

It will be clear from the example that the sum obtained by adding the complement codes is a representation in the complement code of the actual sum of the numbers  $X$  and  $Y$ .

Case 3.  $X > 0; Y < 0$ ; but  $X + Y > 0$

Direct Code	Complement Code	Addition in Compl. Code
$X = 0.101001 \rightarrow$	$[X]_{com} = 0.101001 \rightarrow$	$0.101001$
$+ Y = -0.001101 \rightarrow$	$[Y]_{com} = 1.110011 \rightarrow$	$+ 1.110011$
$X + Y = 0.011100 \rightarrow$	$[X + Y]_{com} = 0.011100$	$10.011100$
	$\uparrow$ $\swarrow$	$\downarrow$ $0.011100$

The carry one obtained in the column of integers after addition in the complement code is rejected, as indicated by the arrow.

Case 4.  $X < 0; Y < 0; X + Y < 0$ .

/32

Direct Code	Complement Code	Addition in Compl. Code
$X = -0.101001 \rightarrow$	$[X]_{com} = 1.010111$	$1.010111$
$+ Y = -0.001101 \rightarrow$	$[Y]_{com} = 1.110011$	$+ 1.110011$
$X + Y = -0.110110 \rightarrow$	$[X + Y]_{com} = 1.001010$	$11.001010$
	$\uparrow$ $\swarrow$	$\downarrow$ $1.001010$

It will be clear from these cases that the addition of numbers in the complement code will also always give the sum in complement code if the above-mentioned definite conditions are observed.

Addition of numbers in inverse code. Inverse codes of numbers are added in the same way as complement codes, place by place, and the sign places are added like the integer places.

A feature of the addition of inverse codes is that a one carried from the sign place (if any) is added to the lowest place of the sum of the codes (the so-called cyclic carry or cyclic shift).

Four cases may occur during the addition of inverse codes.

Case 1.  $X > 0$ ;  $Y > 0$ ; thus also  $X + Y > 0$ .

Since the summands are positive, their representations in the inverse and direct codes are the same, and therefore the representations of the sum in both codes are also the same.

Case 2.  $X > 0$ ;  $Y < 0$ ;  $X + Y < 0$ .

As in the discussion on the various cases of addition of complement codes, we will add the numbers in the direct and inverse codes and compare the writings for the sums so obtained.

Direct Code	Inverse Code	Addition in Inverse Code
$X = 0.001001 \rightarrow$	$[X]_{inv} = 0.001001 \rightarrow$	$0.001001$
$+ Y = -0.110001 \rightarrow$	$[Y]_{inv} = 1.001110 \rightarrow$	$+ 1.001110$
$X + Y = -0.101000 \rightarrow$	$[X + Y]_{inv} = 1.010111 \rightarrow$	$1.010111$

In this case, there is no cyclic carry.

Case 3.  $X > 0$ ;  $Y < 0$ ; but  $X + Y > 0$ .

Direct Code	Inverse Code	Addition in Inverse Code
$X = 0.110001 \rightarrow$	$[X]_{inv} = 0.110001 \rightarrow$	$0.110001$
$+ Y = -0.001001 \rightarrow$	$[Y]_{inv} = 1.110110 \rightarrow$	$+ 1.110110$
$X + Y = 0.101000 \rightarrow$	$[X + Y]_{inv} = 0.101000$	$10.100111$ <div style="display: flex; align-items: center; justify-content: right;"> <div style="border-top: 1px solid black; width: 100px; margin-right: 5px;"></div> <div style="margin-right: 5px;">1</div> <div style="margin-right: 5px;">+</div> </div> $0.101000$

In this case, there is a cyclic carry; the carried one obtained in the integer place, is added to the lowest place of the sum of the inverse codes. /33

Case 4.  $X < 0$ ;  $Y < 0$ ;  $X + Y < 0$ .

Direct Code	Inverse Code	Addition in Inverse Code
$+ X = -0.110001 \rightarrow$	$[X]_{\text{inv}} = 1.001110 \rightarrow$	$1.001110$
$+ Y = -0.001001 \rightarrow$	$[Y]_{\text{inv}} = 1.110110 \rightarrow$	$+ 1.110110$
<hr/>	<hr/>	<hr/>
$X + Y = -0.111010 \rightarrow$	$[X + Y]_{\text{inv}} = 1.000101$	$11.000100$
		$\xrightarrow{\quad} 1 +$
		<hr/>
		$1.000101$

The discussed cases show that if the above definite conditions are observed, the addition of numbers in inverse codes gives a sum which is likewise in inverse code.

Indications of overflow on addition of numbers in complement or inverse code. If, in additions in the complement or inverse code, the sum exceeds unity in absolute value, there will be overflow. This leads to a distortion of the result and is therefore inadmissible in normal operation.

**Example 1.** Overflow on addition of complement codes of numbers.

Direct Code	Complement Code	Addition in Compl. Code
$X = -0.101101$ $+ Y = -0.110101$ <hr/> $X + Y = -1.100010$	$[X]_{\text{com}} = 1.010011$ $[Y]_{\text{com}} = 1.001011$ <hr/> $[X + Y]_{\text{com}}$ does not exist	$+ 1.010011$ $+ 1.001011$ <hr/> $10.011110$

The result of addition in the complement code is 0.011110.

In this example, as a result of the addition, we obtained the complement code of the positive number 0.01111, while the actual sum of the numbers is negative, -1.10001.

**Example 2.** Overflow in addition of inverse codes.

Direct Code	Inverse Code	Addition in Inverse Code
$+X = -0.110010$ $+Y = -0.011111$ <hr/> $X+Y = -1.010001$	$[X]_{inv} = 1.001101$ $[Y]_{inv} = 1.100000$ <hr/> $[X+Y]_{inv}$ does not exist	$+ 1.001101$ $+ 1.100000$ <hr/> $10.101101$ <div style="text-align: right;"> <math>\boxed{10} \rightarrow 1 +</math> </div>

The result of addition in the inverse code is 0.101110. This does not correspond to the actual sum of the numbers.

In case of overflow, the computer must be stopped immediately and the scale factors so changed as to exclude the possibility of further overflow. Obviously, in automatic operation, the computer itself must independently detect overflow and stop. 134

Let us consider how overflow can be determined in addition of numbers in

the complement or inverse codes.

Overflow occurs in cases when the summands X and Y have the same sign.

A detailed study of how overflow takes place will show that the existence of opposite signs of the sum and the summands may serve as a criterion for overflow. If the integral parts of the complement or inverse codes added were units, then in the integral part of the sum we would get a zero and, conversely, if the integral parts of the codes added contain zeros, a one is obtained in the integral part of the sum.

Consequently, to fix the time of overflow, a mechanism for comparing the integral parts of the summand numbers is necessary. If the integral parts coincide, then this mechanism must compare the integral part of the sum with the integral part of one of the summands. If these parts do not coincide, the mechanism would have to issue a control signal to stop the computer. Such a mechanism would be very complex, making this method of detecting overflow inconvenient.

It is less complicated to detect overflow when using a modified complement or inverse code.

Addition of numbers in modified complement code. In a modified complement code the numbers are added in the same way as in the complement code.

Four cases are possible in addition.

Direct Code	Modified Complement Code	Addition in Modified Compl. Code
1) $X = 0.001101$ $Y = 0.110001$ <hr/> $X + Y = 0.111110$	$[X]_{\text{com}}^M = 00.001101$ $[Y]_{\text{com}}^M = 00.110001$ <hr/> $[X + Y]_{\text{com}}^M = 00.111110$	$00.001101$ $00.110001$ + <hr/> $00.111110$
2) $X = 0.110001$ $Y = -0.001101$ <hr/> $X + Y = 0.100100$	$[X]_{\text{com}}^M = 00.110001$ $[Y]_{\text{com}}^M = 11.110011$ <hr/> $[X + Y]_{\text{com}}^M = 00.100100$	$00.110001$ $11.110011$ + <hr/> $100.100100$ <div style="text-align: center;"><math>\uparrow</math> <math>\downarrow</math> <math>00.100100</math></div>
3) $X = 0.001101$ $Y = -0.110001$ <hr/> $X + Y = -0.100100$	$[X]_{\text{com}}^M = 00.001101$ $[Y]_{\text{com}}^M = 11.001111$ <hr/> $[X + Y]_{\text{com}}^M = 11.011100$	$00.001101$ $11.001111$ + <hr/> $11.011100$

$$\begin{array}{rcl}
4) \quad X = -0.001101 & [X]_{\text{com}}^{\text{m}} = 11.110011 & 11.110011 \\
\quad Y = -0.110001 & [Y]_{\text{com}}^{\text{m}} = 11.001111 & 11.001111 + \\
\hline
\quad X + Y = 0.111110 & [X + Y]_{\text{com}}^{\text{m}} = 11.000010 & 111.000010 \\
& & \quad \quad \quad \uparrow \quad \downarrow \\
& & \quad \quad \quad 11.000010
\end{array}$$

Addition of numbers in modified inverse code. In the modified inverse code, numbers are added in the same way as in the ordinary inverse code.

Four cases will be encountered here.

	Direct Code	Modified Inverse Code	Addition in Modified Inverse Code
1)	$X = 0.110001$ $Y = 0.001001$ <hr/> $X + Y = 0.111010$	$[X]_{\text{inv}}^{\text{m}} = 00.110001$ $[Y]_{\text{inv}}^{\text{m}} = 00.001001$ <hr/> $[X + Y]_{\text{inv}}^{\text{m}} = 00.111010$	$00.110001$ $+ 00.001001$ <hr/> $00.111010$
2)	$X = 0.110001$ $Y = -0.001001$ <hr/> $X + Y = 0.101000$	$[X]_{\text{inv}}^{\text{m}} = 00.110001$ $[Y]_{\text{inv}}^{\text{m}} = 11.110110$ <hr/> $[X + Y]_{\text{inv}}^{\text{m}} = 00.101000$	$00.110001$ $+ 11.110110$ <hr/> $100.100111$ $\quad \quad \quad \uparrow$ $\quad \quad \quad 1$ <hr/> $00.101000$
3)	$X = 0.001001$ $Y = -0.110001$ <hr/> $X + Y = -0.101000$	$[X]_{\text{inv}}^{\text{m}} = 00.001001$ $[Y]_{\text{inv}}^{\text{m}} = 11.001110$ <hr/> $[X + Y]_{\text{inv}}^{\text{m}} = 11.010111$	$00.001001$ $+ 11.001110$ <hr/> $11.010111$
4)	$X = -0.001001$ $Y = -0.110001$ <hr/> $X + Y = -0.111010$	$[X]_{\text{inv}}^{\text{m}} = 11.110110$ $[Y]_{\text{inv}}^{\text{m}} = 11.001110$ <hr/> $[X + Y]_{\text{inv}}^{\text{m}} = 11.000101$	$11.110110$ $+ 11.001110$ <hr/> $111.000100$ $\quad \quad \quad \uparrow$ $\quad \quad \quad 1$ <hr/> $11.000101$

Overflow during addition of modified codes. Overflow during addition of modified codes occurs in cases where the sum in absolute value is greater than unity. As in the addition of simple codes, overflow takes place when the summands are of the same sign. Let us consider examples of overflow. /36

Example 1. Overflow during addition of modified complement codes.

$$\begin{array}{rcl}
X = -0.111001 & [X]_{\text{com}}^M = 11.000111 & \\
Y = -0.110100 & [Y]_{\text{com}}^M = 11.001100 & + \begin{array}{r} 11.000111 \\ 11.001100 \\ \hline 110.010011 \end{array} \\
\hline
X + Y = -1.101101 & [X + Y]_{\text{com}}^M \text{ does not exist} & \leftarrow \downarrow
\end{array}$$

Result of addition: 10,010011.

Example 2. Overflow during addition of modified inverse codes.

$$\begin{array}{rcl}
X = 0.110101 & [X]_{\text{inv}}^M = 00.110101 & \\
Y = 0.101101 & [Y]_{\text{inv}}^M = 00.101101 & + \begin{array}{r} 00.110101 \\ 00.101101 \\ \hline 01.100010 \end{array} \\
\hline
X + Y = 1.100010 & [X + Y]_{\text{inv}}^M \text{ does not exist} & \uparrow \\
& & \text{Result of addition}
\end{array}$$

It will be clear from these examples that overflow during the addition of modified codes is manifested in the fact that the result obtained during addition cannot be formulated in any of the modified codes. The criterion of overflow is the disagreement of the digits obtained in the two highest (sign) places.

Thus, in order to detect the instant of overflow the digits in these places must be compared by the aid of a special unit. If the digits disagree, the unit must generate the corresponding control signal.

A unit for overflow control is simpler than for the addition and numbers in unmodified codes, since it performs only the single operation of comparison.

Addition of numbers on floating-point machines. In adding numbers on floating-point computers, the exponents of the summands are first equalized and then the mantissas are added. The exponent of the sum is the total exponent of the summands. The equalization of exponents consists in increasing the lower exponent of the number to a higher exponent with the corresponding change in the mantissa.

The mantissas are usually added in one of the modified codes by the above rules. Three cases can occur. We shall consider them on specific examples. All writings are given in the binary system with respect to the memory location of a floating-point computer (cf. Fig.4). The calculations are performed in modified inverse code. /37

Case 1. Addition proceeds without overflow or disturbance of normalization.

The following entries have been made in the locations of the computer:

Summand  $X$  1 100100110 0 011.  
Summand  $Y$  0 110001101 0 101.



The addition is performed in several stages.

First stage. Shift to the right of summand X (denormalization) to equalize its exponent 011 with the exponent 101 of summand Y. After the shift, the entry of the summand X will be of the form

X 1 001001001 0 101.

Second stage. Conversion of the mantissas of X and Y into modified inverse codes.

Mantissa X 11 110110110.  
Mantissa Y 00 110001101.

Third stage. Addition of mantissas:

$$\begin{array}{r}
 X \ 11 \ 110110110 \\
 Y \ 00 \ 110001101 \ + \\
 \hline
 100 \ 101000011 \\
 \downarrow \quad \quad \rightarrow 1 \ + \\
 \hline
 X + Y \ 00 \ 101000100
 \end{array}$$

Fourth stage. Conversion of the sum X + Y into direct code

0 101000100

and writing out the result:

X + Y 0 101000100 0 101.

Case 2. The addition proceeds without overflow, but the result after conversion to direct code is found to be nonstandardized. This case is called a rightward impairment of normalization.

Normalization is performed before input of the result to the memory location.

The memory cells of the machine have the entries:

Summand X 0 100100110 0 100.  
Summand Y 1 100101010 0 110.

The addition is performed in five stages.

First stage. Equalization of exponents. After shifting, the first ex-

ponent is entered in the cell in the following manner:

$$X \ 0 \ 001001001 \ 0 \ 110.$$

Second stage. Conversion of both mantissas into modified inverse code: /38

$$\begin{array}{r} X \ 00 \ 001001001 \\ Y \ 11 \ 011010101. \end{array}$$

Third stage. Addition of mantissas:

$$\begin{array}{r} X \ 00 \ 001001001 \\ Y \ 11 \ 011010101 \\ \hline X + Y \ 11 \ 100011110 \end{array}$$

Fourth stage. Conversion of the result into direct code:

$$X + Y \ 1 \ 011100001 \ 0 \ 110.$$

Fifth stage. Normalization and entry of the result:

$$X + Y \ 1 \ 111000010 \ 0 \ 101.$$

Case 3. Overflow takes place during addition. To determine the procedure in this case, two facts must be noted:

a) The cause of the overflow is that the sum  $|X + Y| \geq 1$ . Obviously, if after equalization of the exponents the mantissas were shifted one place to the right, there would be no overflow.

b) The criterion of overflow is the presence of different digits in the sign places, the combination 01 indicating that the sum is positive and the combination 10 that it is negative. Consequently, from the left of the two sign places we can immediately establish the sign of the sum (0  $\rightarrow$  "+"; 1  $\rightarrow$  "-").

Starting from these facts, we can immediately obtain the sum without repeated calculations. For this purpose, we must shift the result of the addition one place to the right (and increase the exponent by one) and then introduce into the second sign place the figure standing on the left place. These operations are performed automatically by the computer after the comparator generates a signal indicating the presence of different digits in the sign places. The computer then operates as in the preceding cases.

This case of overflow is customarily called a leftward impairment of normalization.

Example:

X 1 101011011 0 110;  
Y 1 110001101 0 101.

First stage. Shift of second summand to equalize the exponents:

Y 1 011000110 0 110.

Second stage. Translation of the mantissas into modified inverse code:

X 11 010100100;  
Y 11 100111001.

Third stage. Addition of mantissas:

/39

$$\begin{array}{r}
 + X 11 010100100 \\
 + Y 11 100111001 \\
 \hline
 110 111011101 \\
 \begin{array}{c} \text{---} \end{array} \rightarrow 1 + \\
 \hline
 X + Y 10 111011110
 \end{array}$$

There is a leftward impairment of the normalization.

Fourth stage. Shift of the result to the right:

11 011101111.

Fifth stage. Conversion into direct code and write-out of the result:

1 100010000 0 111.

Owing to the rightward shift (fourth stage) the exponent has been increased by one.

## Section 9. Multiplication of Numbers on Computers

On digital computers numbers are multiplied in direct code. The multiplication sign is defined by the addition of numbers designating the signs of the co-factors, and the unit of the highest place (carry) formed on addition is lost.

The addition is performed by the following rule:

$$0 + 0 = 0; 0 + 1 = 1; 1 + 0 = 1; 1 + 1 = 0.$$

The results obtained by such addition are in complete agreement with the well-known product-sign rule of algebra:

$$\begin{aligned}
 (+) \cdot (+) &= (+); (+) \cdot (-) = (-); (-) \cdot (+) = (-); \\
 (-) \cdot (-) &= (+).
 \end{aligned}$$

After determining the sign of the product, the numbers themselves are multiplied.

Multiplication of numbers in fixed-point computers. Numbers represented in the natural form are multiplied by the ordinary rules of arithmetic for binary numbers. The process of multiplication may be accomplished with a shift of the multiplicand to left or right. In the former case, the multiplication begins with the least significant digit of the multiplier, and in the latter case with the most significant digit.

Example:

Leftward Shift  
of Multiplicand

$$\begin{array}{r} \times 1101 \\ 1101 \\ \hline 1101 \\ + 1101 \\ \hline 10001111 \end{array}$$

Rightward Shift  
of Multiplicand

$$\begin{array}{r} \times 1101 \\ 1101 \\ \hline 1101 \\ + 1101 \\ \hline 10001111 \end{array}$$

Let us call the product of the multiplicand by any place of the multiplier a partial product. In multiplying binary numbers, the partial product is equal to the multiplicand (if the corresponding place of the multiplier is one) or equal to zero (if the corresponding place of the multiplier is zero). For this reason, multiplication actually is performed by successive shifts of the multiplicand and addition of the partial products obtained as a result of the shifts.

Multiplication of numbers on floating-point computers. On floating-point computers multiplication of numbers is accomplished in four stages:

- 1) determining the sign of the product;
- 2) determining the exponent of the product by algebraic addition of the exponents of the co-factors;
- 3) multiplying the mantissas of the co-factors;
- 4) normalizing the result (if necessary).

Mantissas are multiplied in the same manner as numbers in fixed-point computers.

The sequence of work in multiplying numbers on floating-point computers will now be illustrated by an example.

Example.  $X = 0.00101101$ ;  $Y = -10000.1$ . Find the products  $XY$ . In the memory cells of the computer the co-factors are entered as follows:

$X \ 0 \ 101101000 \ 1 \ 010$   
 $Y \ 1 \ 100001000 \ 0 \ 101.$

First stage. Determining the sign of the product:

$$0 + 1 = 1.$$

Second stage. Determining the exponent of the product:

$$(-010) + (101) = +011.$$

Third stage. Multiplying the mantissas by the co-factors:

$$\begin{array}{r} \times \quad 0.101101 \\ \quad 0.100001 \\ \hline \quad 101101 \\ \quad \quad 101 \parallel 101 \\ \hline 0.010111001 \parallel 101 \end{array}$$

Since the memory location has nine places for the mantissa, the last three places of the product, set off by the double vertical line, will be lost. The result of multiplication will be of the following form:

$$1 \ 010111001 \ 0 \ 011.$$

Fourth stage. Normalization of the result:

$$1 \ 101110010 \ 0 \ 010.$$

## Section 10. Division of Numbers on Computers

Division is a process inverse to multiplication. If it is possible to multiply numbers by the method of multiple addition, then it is possible to divide them by the method of multiple subtraction. In the binary system this method is very convenient, since the digits of the quotient take only the two values 0 and 1.

Let us consider the general procedure of the work of performing the operation of division by the method of multiple subtraction (for binary numbers).

The digits of the quotient are determined successively, beginning with the most significant digit, by subtracting the divisor from the remainder obtained from the preceding subtraction. In determining the first digit of the quotient, the entire dividend is taken as the remainder.

After each subtraction, the divisor must be shifted to the right relative to the dividend. If the remainder is positive or zero, then the digit of the quotient is 1. If the remainder is negative, then the digit of the quotient is 0. To obtain the next digit of the quotient after a zero, the divisor, shifted an additional place to the right, must be subtracted from the last positive remainder.

### Example.

Dividend	1 1 1 1 0 1 1 1 1	1 0 1 1 0 1	
Divisor	1 0 1 1 0 1 0 0 0	1 0 1 1	quotient
First remainder	1 0 0 0 0 1 1 1		
Second remainder	1 0 1 1 0 1 0 0		
(negative)	- 1 0 1 1 0 1		
Last positive remainder	1 0 0 0 0 1 1 1		
	1 0 1 1 0 1 0		
Third remainder	1 0 1 1 0 1		
	1 0 1 1 0 1		
Fourth remainder	0 0 0 0 0 0		

Division of numbers on fixed-point computers. The sign of the quotient, as in multiplication, is determined by adding the digits forming the signs of the dividend and divisor. On digital computers the operation of subtraction is not performed, and therefore successive subtraction of the divisor is replaced by addition of remainders and the inverse or complement code of the divisor. The remainders are also obtained in the corresponding code.

As we know, in adding complement or inverse codes the adders performing the operation of addition operate under the condition that both the summands and the sum itself do not exceed unity in absolute value. In view of this, the dividend and divisor must be less than unity, while the dividend must be less than the divisor, so that the quotient is likewise obtained less than unity. The necessary values of the dividend and divisors are set up by corresponding selection of the scale factors.

Let the dividend be  $X$ , the divisor  $Y$ , the remainders  $X_1, X_2, X_3, \dots, X_k, \dots, X_n$ , and the digits of the quotient  $Z_1, Z_2, Z_3, \dots, Z_k, \dots, Z_n$ .

The rule of division can be formulated as follows: to determine the  $k$ -th digit  $Z_k$  of the quotient after the point, the divisor  $Y$ , shifted  $k$  places to the right, must be subtracted from the last remainder  $X_{k-1}$ , i.e.,  $2^{-k} Y$  must be subtracted. If the resultant remainder  $X_k = X_{k-1} - 2^{-k} Y$  is positive or zero, then the digit of the quotient  $Z_k = 1$ . If  $X_k$  is negative, then the digit of the quotient  $Z_k = 0$ . To determine the digit  $Z_{k+1}$  of the quotient after a zero, the last positive remainder  $X_{k-1}$  must be restored, for which purpose the divisor  $Y$ , shifted  $k$  spaces to the right, must be added to the remainder  $X_k$ , i.e.,  $X_{k-1} = X_k + 2^{-k} Y$ . The restored positive remainder is taken as the remainder  $X_k$ , and from it  $2^{-(k+1)} Y$  is subtracted, i.e., the divisor is shifted still another place to the right. The division continues similarly thereafter.

To determine the first digit  $Z_1$  of the quotient after the point, the zero-th remainder  $X_0$  is the entire dividend, from which the divisor, shifted one place to the right, is subtracted.

Let us illustrate this rule by an example.

Example.  $X = 0.1001011$ ;  $Y = 0.101001$ . Find  $Z = \frac{X}{Y}$ .

For clarity, all operations will be performed in direct code.

Dividend	:	0.1001011	0.101001 — divisor
Shift divisor to the right	:	101001	0.11101 — quotient
first remainder	:	1000100	
Shift divisor to the right	:	101001	
second remainder	:	0110110	
Shift divisor to the right	:	101001	
third remainder	:	0011010	
Shift divisor to the right	:	101001	
fourth remainder (negative)	:	-01111	
Restored third remainder	:	110100	
Shift divisor to the right	:	101001	
fifth remainder	:	001011.	

Since five shifts were made, the remainder from the division is  $0.001011 \times 10^{-101}$ .

It follows from the above exposition that two cases can be encountered in division, depending on the sign of the remainder  $X_k$ :

1)  $X_k \geq 0$ ; next remainder will be

$$X_{k+1} = X_k - 2^{-(k+1)}Y. \quad (3)$$

2)  $X_k < 0$ ; in this case the preceding remainder must be restored:

$$X_{k-1} = X_k + 2^{-k}Y. \quad (4)$$

Next remainder will be

$$X_{k+1} = X_{k-1} - 2^{-(k+1)}Y. \quad (5)$$

A method exists which, after getting a negative remainder  $X_k$ , permits 43 immediate determination of the next remainder  $X_{k+1}$  without restoring the last remainder  $X_{k-1}$ . Let us derive the formula for this method. In eq.(5), let us substitute the value of  $X_{k-1}$  from eq.(4):

$$\begin{aligned} X_{k+1} &= X_{k-1} - 2^{-(k+1)}Y = X_k + 2^{-k}Y - 2^{-(k+1)}Y = X_k + 2^{-(k+1)}Y; \\ X_{k+1} &= X_k + 2^{-(k+1)}Y. \end{aligned} \quad (6)$$

Thus, after obtaining the remainder  $X_k < 0$ , we can find the next remainder

$X_{k+1}$  from eq.(6) without restoring the last remainder  $X_{k-1}$ . For this purpose, to the negative remainder  $X_k$  we must add the divisor, shifted an additional place to the right. This method is used in modern digital computers.

In accordance with eq.(6), in the example under consideration the last remainder can be found immediately without restoring the third remainder in the following way:

$$\begin{array}{r}
 \text{fourth remainder (negative) } \dots\dots\dots -01111 \\
 \text{Shift divisor one place to the right } \dots\dots\dots 101001 \quad + \\
 \hline
 \text{Fifth remainder } 001011
 \end{array}$$

Depending on the above two cases, the computer may operate in two cycles:

a) If the next remainder  $X_k \geq 0$ , then the next cycle consists in subtracting the divisor shifted one place to the right, from the remainder  $X_k$  [see eq.(3)].

b) If the next remainder  $X_k < 0$ , then the next cycle consists not in subtracting but in adding the divisor shifted one place to the right to the remainder  $X_k$  [see eq.(6)].

In floating-point computers the operation of division is performed in four stages:

- 1) determination of the sign of the quotient;
- 2) determination of the exponent of the quotient;
- 3) determination of the mantissa of the dividend over the mantissa of the divisor;
- 4) normalization of the result (if necessary).

The sign of the quotient is determined in the same manner as in multiplication. For determining the exponent of the quotient, the exponent of the divisor is subtracted from the exponent of the dividend, allowing for their sign.

The mantissas are divided in the same way as in fixed-point computers, but here the dividend need not be less than the divisor. As a result of the division we may obtain a number greater than unity (impairment of normalization to the left) or less than half (impairment of normalization to the right). In these cases the normalization of the result is performed in the same way as is done in the addition of numbers assigned in the normal form.

In order to simplify the arithmetic unit, division in many digital computers is replaced by multiplication by the reciprocal.

/44

Let it be required to determine the quotient  $Z = \frac{k}{X}$ . The computer determines the quantity  $Y = \frac{1}{X}$  and then finds the quotient  $Z = kY$ .



The quantity Y is determined by a special program which will be considered later (see Chapter VIII).

## Section 11. Elements of Mathematical Logic

All the circuits of the control and computing units of electronic digital computers are built of elements that perform logic operations. Such elements are customarily called logic elements.

Logic in the general sense is the science of the canons and criteria of validity in thought, while symbolic or mathematical logic is the science that deals with the application of mathematical methods to the solution of various logical problems. In digital computers the initial part of mathematical logic is mainly used. This is the calculus of propositions, or logical algebra. In the foreign literature, logical algebra is often called Boolean algebra, from the name of George Boole, who developed many propositions of mathematical logics in the second half of the last century. Mathematical logic began to find practical application only in the 1940's. At this time there appeared complex switching circuits in automatic telephone systems and in computing mechanisms; to calculate these circuits the apparatus of mathematical logic had to be called upon. Today mathematical logic is widely used in the design and analysis of various logic circuits, as well as in the description of their operation.

The concept of the proposition and its truth value. The subject of logical algebra is what are called propositions.

A proposition is any assertion that can be said to be either true or false, for example, snow is white, water is hard, it is now 9 AM, etc. Propositions are evaluated only from the point of view of their truth or falsehood. No other criteria (good proposition, bad proposition, etc.) are considered in logical algebra. A proposition cannot be simultaneously both true and false.

Let us define still another idea, the truth value of a proposition. If a proposition is true, it is said that its truth value is one, if a proposition is false, then its truth value is zero. Two propositions are called equivalent if their truth values are the same.

Thus the truth value of a proposition is a variable quantity which can take only two discrete values, zero or one, like the digits in the binary 45 system of notation.

This makes it possible to apply the binary system to the calculation of the truth values of various propositions. This remarkable property of the binary system has been responsible for its widespread use in electronic digital computers.

Propositions may be simple or compound. A compound proposition is obtained by combining simple propositions by the aid of so-called logic operators.

Logical algebra is concerned with establishing the relationship between

simple and compound propositions, mainly by solving the following problems:

Finding the truth values of a compound proposition as a function of the truth values of the simple propositions of which it is composed.

Finding the truth values of the simple propositions that make up a compound proposition as a function of the truth value of that compound proposition.

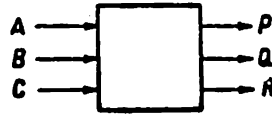


Fig.7 General Symbolic Designation of a Logic Element

Let us denote simple propositions by the first letters of the Latin alphabet (A, B, C, D) and compound propositions by the last letters (P, Q, R, S).

The equivalence of two propositions is denoted by the equality sign, and the truth value of any proposition by the same letter as the proposition itself. For example the notation  $A = 1$  means that the proposition A is true, i.e., the truth value of the proposition A is unity. The notation  $B = 0$  means that the proposition B is false, i.e., its truth value is zero. The notation  $A = C$  means that the propositions A and C are equivalent.

The logic elements of electronic digital computers representing logic operators are constructed of various electrical circuits. In this case, simple propositions are represented by signals arriving at the input of the circuit and complex propositions by the output signals. If there is a signal, then the truth value of the proposition it represents is unity; if there is no signal, then the truth value of the proposition represented by the signal is zero. Since a logic element of a computer represents a specific logic operator, a strictly determinate combination of input signals corresponding to the truth values of simple propositions must correspond to the presence (or absence) of a signal at the output of a circuit. The realization of logic operators inside the circuit is accomplished by the aid of switching circuits.

Figure 7 shows the conventional representation of a logic element. The signals A, B, C, representing simple propositions, are applied to the input of the element.

In the general case, a logic element may have several inputs and outputs (P, Q, R...). This means that its circuit realizes several logic operators. To a definite combination of signals at the inputs of the elements there corresponds a definite combination of signals at their outputs. /46

Let us consider the logic operators most often encountered and the compound propositions corresponding to them.

Logic operators. The first of the three operators considered below are the fundamental logic operators.



Fig.8 Conventional Designation of the  
Logic Element NOT (Inverter)

1. Negation (logic operator NOT). This operator signifies the negation of the original proposition. Negation is denoted by a macron above the designation of the proposition:  $\bar{A}$  (read "not A"). The sign of the logic operator "-" is read "not".

TABLE 5

TABLE OF THE LOGIC OPERATOR NOT  
(NEGATION)

$A$	0	1
$P = \bar{A}$	1	0

The negation of the proposition A is the term applied to the compound proposition P, which is true if the proposition A is false and false if the proposition A is true.

The logic operator NOT is written by the following formula:

$$P = \bar{A} \text{ ( read : „P is not A“ ).}$$

The same logic operator can be illustrated by a Table showing the truth values of the compound proposition P as a function of the truth values of the simple proposition A composing it (Table 5).

The logic operator "negation" is realized in electrical circuits by the logic element NOT. As applied to electrical circuits, this operator means that a signal appears at the output of the circuit if there is no signal at its input, and conversely, a signal will be absent at the output of the circuit if there is a signal at its input.

The logic element NOT in the circuit of a digital computer is often called an inverter. The symbolic designation of an inverter is shown in Fig.8.

2. Logical multiplication (the logic operator AND). The logical operator AND is symbolically denoted by the multiplication sign:

$$P = A \cdot B \text{ ( read : „} P \text{ is } A \text{ and } B \text{“).}$$

A coupling between simple propositions such that a compound proposition /47 is true only when all the constituent simple propositions are true is called logical multiplication. This coupling is illustrated in Table 6.

TABLE 6

TABLE OF THE LOGIC OPERATOR AND  
(LOGICAL MULTIPLICATION)

<i>A</i>	0	1	0	1
<i>B</i>	0	0	1	1
$P = A \cdot B$	0	0	0	1

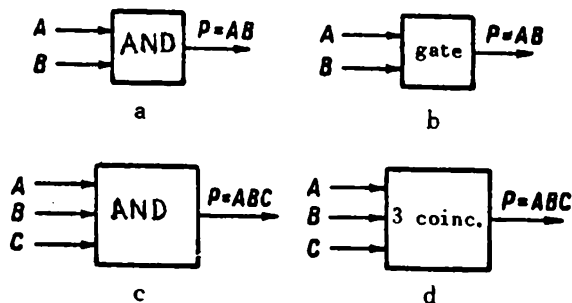


Fig.9 Symbolic Denotations of the Logic Element AND:  
a - At two inputs; b - At two inputs (gate); c - At  
three inputs; d - At three inputs (coin-  
cidence circuit)

The logic operator AND is sometimes called conjunction and is symbolically denoted by the following symbols:

$$P = A \wedge B; \quad P = A \& B.$$

We shall denote the operator AND by the multiplication sign.

In the electrical circuits realizing the logic operator AND, a signal appears at the output of the circuit only if there are signals simultaneously at all its inputs.

A circuit reproducing the operation of logical multiplication has been given various designations, of which the following are the most frequent:

logic circuit AND;  
coincidence circuit;  
gate (AND circuit with two inputs).

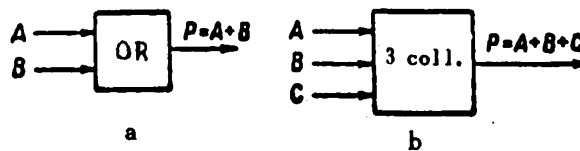


Fig.10 Conventional Symbols for the Logic Element OR  
a - At two inputs; b - collector circuit at three inputs

Figure 9 shows various designations of the logic element AND encountered in circuits.

TABLE 7

TABLE OF THE LOGIC OPERATOR OR  
(LOGICAL ADDITION)

<i>A</i>	0	1	0	1
<i>B</i>	0	0	1	1
$P = A + B$	0	1	1	1

3. Logical addition (logic operator OR). The logic operator OR is symbolically represented by the addition sign: 48

$$P = A + B \text{ (read : „} P \text{ is } A \text{ or } B \text{“).}$$

Logical addition denotes the relation between simple propositions such that the compound proposition is true if and only if at least one of its component simple propositions is true, and false if and only if all these simple propositions are false (Table 7).

The logic operator OR is also called disjunction and is symbolically denoted by the symbol:

$$P = A \vee B.$$

We shall use the former symbol ( $P = A + B$ ).

In electrical circuits realizing the logic operator OR, a signal appears at the output if and only if a signal is fed to at least one of the circuit inputs.

A circuit reproducing the operation of logical addition is usually called an OR gate or a collector circuit. It is sometimes termed a disjunctive circuit.

Figure 10 gives versions of the symbolic representation of the logic element OR.

TABLE 8

TABLE OF EQUIVALENCE OF TWO PROPOSITIONS

<i>A</i>	0	1	0	1
<i>B</i>	0	0	1	1
$P = A \sim B$	1	0	0	1

Elements realizing the basic logic operators NOT, AND, and OR are called basic logic elements in digital computers.

4. Equivalence of two propositions. This operator is denoted by the symbol " $\sim$ ". The notation  $A \sim B$  is read: "A is equivalent to B". /49

The equivalence of two propositions A and B is the compound proposition P, which is true if and only if the simple propositions of which it is composed are simultaneously either both true or both false (Table 8).

The formula for equivalence is as follows:

$P = A \sim B$  (read: "The truth of the proposition P is the truth of the proposition that A is equivalent to B").

The logic operator of equivalence permits us to obtain another logic operator "negation of equivalence" which is widely used in electronic digital computers.

5. Negation of the equivalence of two propositions. This operator is obtained by the aid of two operators already considered by us: negation and equivalence. It is written  $\overline{A \sim B}$  (the equivalence of A and B is negated). For convenience, a special sign has been introduced to represent the negation of equivalence; using it, this operator will appear as follows:

$P = A \approx B$  (read: "The truth of the proposition P is the truth that A is not equivalent to B").

The negation of equivalence is a compound proposition, which is true only when the simple propositions of which it is composed have opposite truth values (Table 9).

TABLE 9

TABLE OF NEGATION OF EQUIVALENCE

<b>A</b>	0	1	0	1
<b>B</b>	0	0	1	1
<b><math>P = A \approx B</math></b>	0	1	1	0

The electrical circuit realizing the operation of negation of equivalence operates as follows: A signal appears at the output of the circuit when there is a signal on only one of its inputs; there is no signal at the output if there are simultaneously either signals or no signals at its inputs. /50

The logic circuit reproducing the operation of negation of equivalence is called a comparator circuit (it is also called an OR - OR circuit or a non-equivalence circuit).

Its function actually is to compare the two quantities fed to its input. If a signal is applied to one input but not to the other, this means that different values have appeared at the inputs of the circuit. A signal appearing at the output will indicate this. If the same values arrive at the input of the circuit, there will be no signal at the output.

Basic laws of logical algebra. There are four fundamental laws of logical algebra: commutative law, associative law, distributive law, and the law of inversion.

1. Commutative law:

$$\text{for addition} \quad A + B = B + A; \quad (7)$$

$$\text{for multiplication} \quad A \cdot B = B \cdot A. \quad (8)$$

2. Associative law:

$$\text{for addition} \quad (A + B) + C = A + (B + C); \quad (9)$$

$$\text{for multiplication} \quad (AB)C = A(BC). \quad (10)$$

3. Distributive law:

$$\text{for addition} \quad (A + B)C = AC + BC; \quad (11)$$

$$\text{for multiplication} \quad AB + C = (A + C)(B + C). \quad (12)$$

4. Law of inversion:

$$\text{for addition} \quad \overline{A + B} = \overline{A} \cdot \overline{B}; \quad (13)$$

$$\text{for multiplication} \quad \overline{A \cdot B} = \overline{A} + \overline{B}. \quad (14)$$

The commutative and associative laws, as well as the distributive law for addition, are encountered in ordinary algebra and require no special proof. The distributive law of multiplications and the law of inversion do not exist in ordinary algebra. We will prove the validity of the law (12) by the aid of Tables of truth values of the compound logical relations described by expression (12). If the truth values for the right and left sides of this expression coincide, the law will be proved.

To construct a Table proving the validity of the law (12), we must set up all possible combinations of the truth values of the propositions A, B, and C. There are eight such combinations, to which eight columns of the Table correspond. We shall also find the truth values of the compound propositions composed of the propositions A, B, and C (Table 10). /51

A comparison of the starred lines will show that they are the same. Consequently, the validity of eq.(12) has been proved. The validity of the law of inversion can be proved in a similar manner.

Transformation of logical expressions. In solving various problems connected with the development or analysis of the circuits of digital computers, very complex logic functions may result. If the logic circuits are constructed directly from such functions, they will be extremely unwieldy. In many cases, however, by making use of the fundamental laws of logical algebra and the resultant consequences, complex logic functions can be substantially simplified.



It is proved in logical algebra that any compound proposition can be constructed from the three logic operators NOT, AND, and OR. For this reason, such operators are called basic operators, and the logic elements realizing them are called basic logic elements.

TABLE 10

PROOF OF VALIDITY OF THE DISTRIBUTIVE LAW FOR  
LOGICAL MULTIPLICATION

A	0	1	0	1	0	1	0	1
B	0	0	1	1	0	0	1	1
C	0	0	0	0	1	1	1	1
A · B	0	0	0	1	0	0	0	1
AB + C	0	0	0	1	1	1	1	1
A + C	0	1	0	1	1	1	1	1
B + C	0	0	1	1	1	1	1	1
(A + C)(B + C)	0	0	0	1	1	1	1	1

The simplification of complex logic functions can be accomplished by transforming the expressions, so as to decrease the number of basic logic operators. This leads to a decrease in the number of basic logic elements realizing the complex logic functions and thus to a simplification of the entire logic circuit.

Some relations can be simplified by making use of the fundamental laws of logical algebra.

Given the expression  $P = AC + BC$ . This expression includes two AND operators and one OR operator. Making use of eq.(11), the relation P can be simplified by removing C from the parentheses: /52

$$P = (A + B)C.$$

This expression is simpler, since it contains one AND operator and one OR operator.

Making use of the properties of the basic logic operators and the laws of logical algebra, a series of propositions can be set up which will help considerably in simplifying compound logical expressions. These propositions are presented below.

Let us first consider the so-called always true and always false propositions, i.e., compound propositions such that they remain true or false regardless of the truth values of the simple propositions composing them.

Always-true propositions:

$$A + 1 = 1. \quad (15)$$

The validity of this equality results from the properties of logical addition,

$$A + \bar{A} = 1. \quad (16)$$

This equality can easily be reduced to the preceding one, if specific truth values (0 or 1) are taken for the proposition A.

Always-false propositions:

$$A \cdot 0 = 0. \quad (17)$$

The validity of this proposition results from the definition of logical multiplication,

$$A \cdot \bar{A} = 0. \quad (18)$$

This equality can easily be reduced to the preceding one by substituting specific truth values for the proposition A.

Other expressions that can be used to simplify compound logical propositions:

$$\bar{\bar{A}} = A, \quad (19)$$

i.e., a double negation is equivalent to affirmation. In fact,

$$\begin{aligned} \bar{\bar{1}} = \overline{(\bar{1})} = \bar{0} = 1; \quad \bar{\bar{0}} = \overline{(\bar{0})} = \bar{1} = 0. \\ A \cdot A \cdot A \cdot \dots \cdot A = A. \end{aligned} \quad (20)$$

The equality follows from the definition of logical multiplication

$$A + A + A + \dots + A = A. \quad (21)$$

The equality results from the definition of logical addition

$$A \cdot 1 = A. \quad (22)$$

Actually, the truth value of this proposition equals the truth value of 53 the proposition A, since logical multiplication takes place here:

$$A + 0 = A. \quad (23)$$

Also for this proposition, the truth value is equal to the truth value of the proposition  $A$ ,

$$A + AB = A. \quad (24)$$

Let us remove  $A$  from the parentheses on the basis of the law (11). Bearing eqs.(15) and (22) in mind, we obtain

$$\begin{aligned} A + AB &= A(1 + B) = A \cdot 1 = A. \\ A + AB + AC &= A. \end{aligned} \quad (25)$$

The proof is the same as for eq.(24),

$$A(A + B) = A. \quad (26)$$

Proof:

$$\begin{aligned} A(A + B) &= AA + AB = A + AB = A(1 + B) = A \cdot 1 = A. \\ A(A + B)(A + C) &= A. \end{aligned} \quad (27)$$

This is proved in the same way as in the preceding case,

$$A + \bar{A}B = A + B. \quad (28)$$

Based on the law (12), we may write

$$\begin{aligned} A + \bar{A}B &= (A + \bar{A})(A + B) = 1 \cdot (A + B) = A + B. \\ (A + B)(A + C)(B + C) &= AB + AC + BC. \end{aligned} \quad (29)$$

We note likewise that

$$A \sim B = (A + \bar{B})(\bar{A} + B). \quad (30)$$

$$A \approx B = A\bar{B} + \bar{A}B. \quad (31)$$

The validity of the last two expressions is proved by the aid of truth Tables, just as the validity of eq.(12) was proved.

## GENERAL PRINCIPLES OF DIGITAL COMPUTER DESIGN

Section 12. Types of Digital Computers

There are many kinds of digital computers. The features in which they differ are as follows:

- volume and character of the problems solvable by the computer, i.e., its arithmetic and logic capabilities;
- purpose of the computer;
- technical and physical principles on which operation of the main units of the computer is based;
- design features and structural layout of the computer.

As to volume and character of the solvable problems, digital computers are divided into two groups, general-purpose and special-purpose.

General-purpose computers are designed to solve complex and varied problems from the widely differing fields of science and technology - mathematics, physics, astronomy, geodetics, automatic regulation and control theory, strength of materials theory, meteorology, etc. All general-purpose computers have a program control, permitting their use for solving practically any group of problems whose character need not be known in designing the computer. The sequence of mathematical logical operations, i.e., the operational program, is set up for each specific problem and fed to the computer before starting the computations. The computer then automatically performs the calculations in accordance with the assigned program.

Special-purpose computers are designed to solve a narrow range of problems or a single problem in some specific field of science or technology. Such machines solve problems of the same character, with different input data. In the design of such computers the volume and character of the problems to be solved is established in advance. For this, the operational program of the computer is handled by appropriate switching of electrical circuits and cannot be modified. Thus, computers with a rigidly fixed program control are called special-purpose computers.

General-purpose computers are divided into large, medium, and small machines. There is no basic difference between these types. Large computers are characterized by high speed of computation, performing tens and hundreds of thousands, and even millions, of arithmetic operations in one second. They have a wide range of representation of numbers (40 - 50 binary places) and high arithmetic and logic capabilities. These computers have complex hardware, contain several thousand electron tubes and semiconductor elements, occupy a floor space of 200 - 300 m<sup>2</sup>, and draw 100 - 150 kw electric power. /55

Small computers operate at lower speed (thousands or tens of thousands of operations per second) and have a narrower range of numbers representation.

These machines are simpler in design, occupy a floor space of 40 - 60 m<sup>2</sup> and draw 10 - 15 kw electric power. Medium-size general-purpose computers occupy a position intermediate between the large and small machines.

All general-purpose computers are calculating machines, i.e., they are designed to perform computational work.

So-called information-logic machines have recently entered the development stage, on the basis of the general-purpose computer. In these, the hardware for input, output, and data storage units is more complex than in general-purpose machines, and they are designed for the collection, analysis, and storage of a large quantity of varied information, which may amount to tens and even hundreds of millions of numerical data.

Special-purpose computers may be of the computational or controlling type. The special-purpose calculating machines are divided into two groups:

- computers to solve special problems in military technology, mathematics, physics, meteorology, etc.;
- computers for automation of accounting and planning work.

Computers of the first group can solve very complex computational problems.

Computers of the second group are used for calculation connected with accounting, planning, supply, and statistics. These computers are characterized by a very large amount of input data and relatively simple calculation.

Control digital computers are designed to control some objects or production processes, for example automatic control of machine tools and production lines or automatation of train runs. Control machines are particularly important in military technology; they are useful in automatic fire control for moving targets, in guidance of rockets, in piloting aircraft, etc. 156

By comparison with general-purpose computers, special-purpose computers are simpler in design, lower in weight, smaller in size, and far cheaper to produce.

Figure 11 gives a flow sheet for classification of digital computers according to problems to be solved and purpose.

Depending on the technological and physical operating principles, the units in digital computers may be:

- mechanical;
- electrical;
- electromechanical;
- magnetic;
- electronic.

The principal units of digital computers are as a rule of the electronic type, operating in conjunction with semiconductor and ferromagnetic elements; the auxiliary units are mostly electrical and electromechanical.

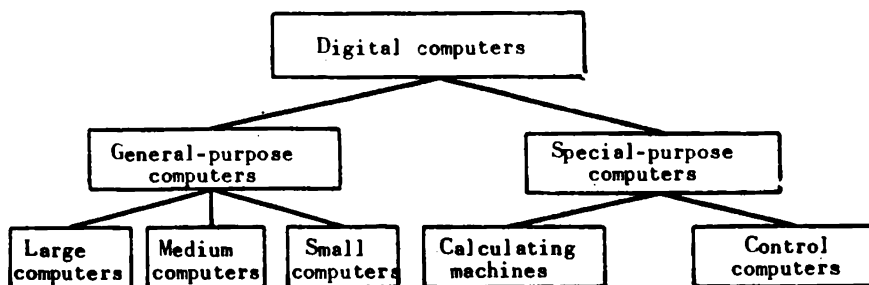


Fig.11 Types of Digital Computers

Depending on design and structural features, digital computers can be classified in accordance with several criteria:

systems of notation used: binary, binary-coded decimal, and decimal;  
 methods of numbers representation: fixed-point and floating-point computers;  
 number of addresses in the command: one-address, two-address, three-address, and four-address computers (see Section 14);  
 operating principle of the electronic units: potential and pulse computers;  
 character of transfer of the numerical data: parallel-operation, serial-operation, and parallel-serial operation machines (see Section 15);  
 type of operating cycles: constant-cycle and variable-cycle computers (see Section 14).

All types of digital computers are characterized by these features.

Table 11 gives the principal characteristics of several Soviet electronic digital computers. 157

### Section 13. Principal Units of an Automatic Digital Computer

To explain the purpose and interaction of the principal units of an automatic digital computer, let us establish an analogy between calculations performed by hand, i.e., by the aid of a pencil and paper, and calculations automatically performed by a computer.

Computations are performed by hand by a human computer using the following

TABLE 11

PRINCIPAL CHARACTERISTICS OF SEVERAL MODELS OF SOVIET  
ELECTRONIC DIGITAL COMPUTERS

Designation of Computers	Computer Type	System of Notation	Number of Binary Places	Average Speed (Number of Operations, sec)	Capacity of Operative Memory Unit (Number of Locations)	Number of tubes and Transistors	Power Drawn, kw	Space Occupied, m <sup>2</sup>
BESM-2	Large, general-purpose, floating-point, three-address, parallel-operation	Binary	39			4000 tubes, 5000 diodes	75	100
"Strela"	Same	Same	43			8000 tubes, 60,000 diodes	120	300
"Ural-1"	Small, general-purpose, fixed-point, single-address, serial-parallel operation	Same	36	100	1024	800 tubes, 3000 diodes	8	60
"Ural-2"	Small, general-purpose, fixed-point, single-address, parallel-operation	Same	40			2500 tubes, 14,000 transistors	25	90
"Dnepr"	Small, general-purpose, fixed-point, two-address, parallel-operation	Same	26		1 unit: 512, in all up to four units	Transistor-operated	1.5	50
"Minsk"	Same	Same	31		1024	800 tubes, 51 transistors	14	40
"Minsk-2"	Small, general purpose, fixed-point, two-address, parallel action	Same	37		1 unit: 4096; only one or two units	Transistorized	4	50

aids:

computation blank;  
adding machine (desk calculator);  
trigonometric and other function Tables.

Figure 12 is a block diagram for the connections between the aids used by a human computer in his calculations. The solid lines indicate the connections over which the numerical data are transferred, and the broken lines indicate the control connections.

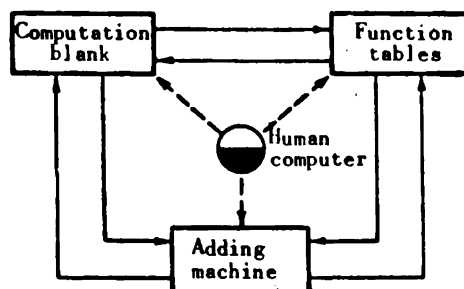


Fig.12 Block Diagram of Connections in Manual Computations

Before beginning work, the human computer analyzes the formula according to which the calculations are to be made. He then selects the numerical method of solution and establishes the required sequence of arithmetic operations which he notes on the computation blank. He also enters the initial data for the calculations on that blank.

Only then can the calculations begin.

From the computation blank the human computer takes two initial numbers, enters them into the adding machine, and by means of this machine performs the arithmetic operation prescribed on the computation blank. The result of the operation is again entered on the blank. If necessary, for the arguments taken from the computation blank or obtained on the adding machine, he finds in the Table the value of the required function and enters it into the machine. Computation blank and Tables are used for expanding the memory of the human computer. The adding machine facilitates and accelerates the performance of the arithmetic operations.

The human computer controls the entire computational process. He checks on accuracy of the required sequence of operations and analyzes the results obtained. Depending on the intermediate results, he modifies the computational procedure if this had been prescribed in advance. He also determines the time of stopping the calculations. An operation on two numbers takes an average of 40 - 50 sec. Of this time, 5 to 8 sec is taken up by operation of the adding



machine (an electric keyboard calculator), while the rest of the time is consumed in selecting the original numbers, entering them into the machine, and writing out the results on the computation blank. All these operations are performed by the human computer. Thus, the first step toward increasing the speed of calculations consists in eliminating the human being from the computational process. The second step is increasing the speed of operation of the calculating machine itself. /59

Modern electronic computing machines perform calculations without human participation. The human being only supervises their operation. The speed of operation of such machines is measured in thousands, tens of thousands, and even hundreds of thousands of operations per second.

For the machine to automatically perform work analogous to the performance in manual calculation, it must have three units:

- arithmetic unit;
- memory unit;
- control unit.

The arithmetic unit plays the same role as the adding machine in hand calculations except that it operates incomparably faster.

The memory unit has a function similar to the function of the computation blank and the Tables. It serves for input, storage, and output of the initial data of the problem, the tabular material, the intermediate and final results of the computation. The memory unit also stores the operating program of the machine.

The control unit controls the computational process by furnishing the sequence for performance of the operations prescribed by the program, i.e., it plays the role of the human being.

In addition to the above basic units, the machine also has auxiliary units:

- a unit for input of the initial data and program into the machine;
- a unit for output of the results of the computations;
- a control desk or console.

The input unit serves to transform the input quantities into the electrical signals that have been adopted for the representation of numbers in the machine. The output unit is designed to put out the results of the calculations in the form of printed signs or other codes convenient for further utilization. The control desk serves for supervising the course of the computation and for operative control of the machine (start, stop, checking individual units, etc.).

#### Section 14. Structural Diagram of the Automatic Digital Computer

The structural layout of any automatic digital computer includes the above basic and auxiliary units. Figure 13 is a generalized structural diagram of an

automatic computer.

This diagram shows six units: arithmetic AU, memory MU, control CU, in- 60  
put IU, output OU, and manual control console MCC.

The arrows indicate the directions of transfer of the numbers, commands, and control signals.

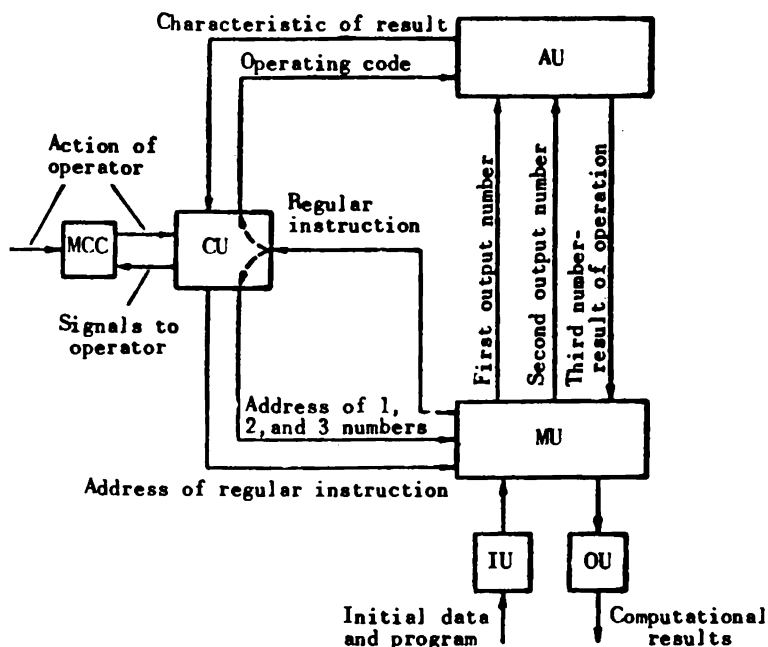


Fig.13 Structural Block Diagram of an Automatic Digital Computer

Let us consider the interaction of the basic units of the machine during the course of computation. The automatic operation of the computer consists in successive performance of the operations prescribed by the program. Each operation is performed under the action of a special control instruction signal. The system of instructions constitutes the operating program of the computer.

The operating program and the initial data are coded by digits and fed over the input unit to the memory unit of the computer. From here on, all calculations are performed automatically by the computer.

The memory unit consists of separate cells or locations. Each of these is designed to store one number or one instruction. All the locations or registers are serially numbered, and each has a permanent number which is known as the address of the location. The address of a location is at the same time the address of the number or instruction stored in it.

An instruction or command is a set of numbers divided into several parts.

One part of the instruction is called the operation code. The operation /61 code determines the arithmetic or logic operation to be performed on the initial numbers. The remaining parts of the instruction are called addresses. The addresses indicate the number of the registers of the memory unit in which the initial numbers are stored, as well as the number of the register into which the result of the operation on these numbers is to be entered.

Example of entry of the formula for an instruction:

01      3206      0101      2507,

where 01 is the operation code;

3206 is the address of the first initial number;

0101 is the address of the second initial number;

2507 is the address of the third number (the address of the register into which the result of the operation on the first and second numbers is to be fed).

Assume that 01 is the operation code of addition. Then the above instruction is interpreted to mean: "Add the number in register No.3206 to the number stored in register No.0101, and enter the results of the addition into register No.2507".

The instructions are entered on the program blank in the octal system of notation, and in the memory unit in the binary system of notation.

The above example was for a three-address instruction. Depending on the computer design, various numbers of addresses may go into an instruction. There are one-, two-, three-, and four-address instructions in existence. Computers are accordingly called one-, two-, three-, and four-address computers.

There are two types of computers, according to the methods of carrying out the instructions of the operating program: with natural sequence of execution of the instructions and with forced sequence.

In computers of the first type all instructions are fed to the registers of the memory unit in the order of their execution. The registers that store these instructions have addresses increasing in the order of their numbers.

After execution of the instruction entered in the next register, the computer proceeds to carry out the instructions stored in the register with the number greater by one. This method of operating is continued to the end of the program or until an instruction is given that may modify the sequence of execution of the calculations.

One-address and three-address computers operate on the natural sequence of execution of instructions.

In computers of the second type the routine instruction indicates the

address of the register where the next instruction is stored. Four-address computers have a prescribed sequence for execution of instructions; the number of the register where the next instruction is stored is indicated in the fourth address. /62

Each instruction (one arithmetic or logic operation) is executed in the course of a single operating cycle of the computer. The entire operating cycle can be divided into three stages.

First stage. The control unit sends the address of the next instruction to the memory unit. The instruction extracted from this address is again fed into the control unit, which divides it into two parts: The operation code is sent to the arithmetic unit, while the address is sent to the memory unit. The arithmetic unit proceeds to perform the operation defined in the code received by it. The memory unit proceeds to put out the first initial number from the register whose number is indicated in the first address of the instruction, then the second initial number from the register whose number is indicated in the second address. The two numbers next arrive at the arithmetic unit. Simultaneously, the memory unit prepares to receive the numbers in the register indicated in the third address of the instruction.

Second stage. The arithmetic unit performs the required operation. This stage takes 50 - 70% of the entire working cycle of the computer.

Third stage. The result of the operation extracted from the arithmetic unit is entered in the register indicated in the third address. At the same time, the control unit generates the address of the instruction that will be executed in the following cycle.

During the course of computation it often becomes necessary to change the sequence of calculations, depending on the intermediate results. In manual calculations, this is done by the human computer himself. The machine, conversely, automatically changes the routing of the computation without human intervention. This is done by a special signal "result criterion" which is generated in the arithmetic unit in the third stage of the working cycle and is fed to the control unit. The character of the signal depends on the value and sign of the result. If necessary, depending on the signal of the result criterion, the control unit may change over to execution of a different part of the program by a special transfer or jump instruction. Naturally, variants that may be encountered during the calculations must be provided for in advance in the program.

The arithmetic unit is the most important unit of the computer and determines its speed. The speed of a general-purpose digital computer means the number of arithmetic (or logic) operations that the computer can perform in one second. The speed of a special-purpose control computer means the time required to solve the entire problem and put out the result, measured from the time the next group of initial data arrives at the input.

Depending on the required speed, arithmetic units may be designed in different ways. /63

When high speed is required, an arithmetic unit is designed on the modular concept, each module being designed to execute a definite arithmetic operation (addition, multiplication, or division). The resultant unit, however, is highly intricate and of considerable size and weight.

When low weight, small dimensions, and simple design are more important than speed, the arithmetic units are designed to contain only an adder and a shift unit. In this case, the operations of multiplication and division are replaced by addition (subtraction) with a shift. Such arithmetic units are of simple design, are small in size and low in weight, but their speed is relatively low.

The memory units of digital machines must meet two requirements: high speed (high speed of intake and output of numbers) and large capacity (ability to store many numbers simultaneously).

The first requirement is based on the fact that the memory and arithmetic unit operate simultaneously, and that the speed of the entire computer depends on the speed of these units. The second requirement results from the necessity of simultaneously storing a very large amount of varied information during the computation process. Up to now it has been impossible to design memory units that fully satisfy both requirements. The memory units of digital machines therefore usually consists of two units, an internal and an external. The internal memory is often called the operative memory unit (OMU) or the machine memory. The external memory unit is sometimes called the storage or store.

The OMU, characterized by high speed, is directly connected to the arithmetic unit. The unit is used to store the numbers and programs required for the immediate computational process. The intermediate calculation results are also entered here. The capacity of such a unit is relatively small (1024-4096 registers). It is usually composed of elements ensuring high-speed intake and output of numbers.

The external memory unit may be of practically unlimited capacity (tens and hundreds of thousands of numbers) but has a far lower operating speed than the internal memory unit. This unit is not directly connected with the arithmetic unit but only over the OMU. During the computational process, information is exchanged between the internal and external memory units. The external memory units consist of magnetic tapes and drums.

Some digital computers, for instance control computers, may have no external memory units.

The design and composition of a control unit is determined by the purposes of machine and the features of its arithmetic and memory units. /64

The control unit of a general-purpose computer usually consists of several registers or modules, each of which controls the work of the basic AU or MU. There is also a central register coordinating the action of all the CU registers and controlling the operation of the auxiliary units of the computer. In modular designs, the control units of digital computers operate with a variable

working cycle. The duration of a cycle is determined by the duration of the operation performed by the arithmetic unit.

The operations of addition and subtraction are executed most rapidly of all the arithmetic operations. The operation of multiplication takes about twice as long, and the operation of division five to ten times as long.

In some types of special-purpose computers, the control unit is a single module. Such computers operate on the constant working cycle whose duration is determined by the duration of the most complex operation.

Control units of modular design operating on a variable cycle, give a higher speed but are more complex in design.

The data input and data output units are often called external units. The external units of a general-purpose computer operate far more slowly than the computer itself, since they are mechanisms of the electromechanical type. In view of this fact, a general-purpose computer as a rule has several sets of such units operating in parallel, i.e., preparing the initial data and formulating the results immediately after every few problems. The initial data and the program are first entered on paper in numerical form. They are then transferred by the operator onto punch cards, punched tape, or magnetic tape, and are then transferred from these over the input unit to the memory unit. The results are likewise put out on punch cards, punched tapes, or on printers which print the results in tabulated form.

The input unit of such computers transforms the continuously varying initial quantities into their digital equivalents for certain definite instants of time. An example of such a device is a shaft-position encoder or voltage comparison converter for converting these data into their numerical equivalents (mechanisms of the "shaft-digitizer" and "voltage-digitizer" type).

The output units of control computers continuously convert the results obtained in numerical form into the corresponding continuous quantities of angles of rotation or voltages (mechanisms of the "digital-to-shaft" and /65 "digital-to-voltage" converter type).

More detailed information on the units of digital computers is presented in the following Chapters, where circuits of these devices are discussed and their operation described.

## Section 15. Representation of Binary Numbers on Digital Computers

All information used in the computation is represented in computers in the form of numbers in some system of notation. The binary system is ordinarily used.

To represent the places of binary numbers in computers we must physically realize two different signals, one of which must correspond to the representation of one and the other to the representation of zero. For example, on punch

cards and punched tapes the binary digits are represented mechanically, by punching of holes. The presence of a hole at a definite place denotes a one, its absence a zero.

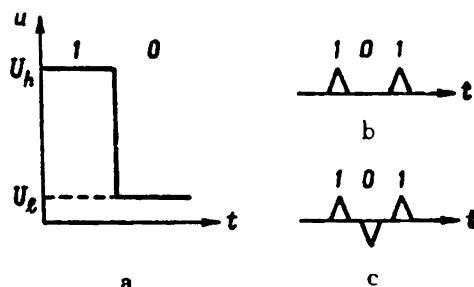


Fig.14 Representation of Binary Numbers in Digital Computers  
 a - Static method (potential code); b, c - Dynamic method (pulse code)

Representation of binary numbers. To represent binary numbers on electronic digital computers, two methods can be used: static (potential code) and dynamic (pulse code).

In the static method, the binary digits are represented by different voltage levels. Usually the high voltage level  $U_h$  is used to represent a one and

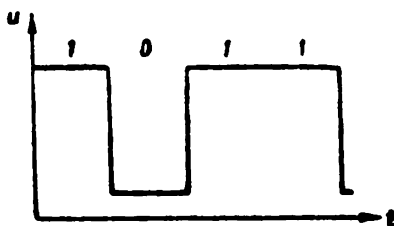


Fig.15 Transfer of a Number in Serial Code with the Digits Represented by the Static Method

the low voltage  $U_l$  to represent a zero (Fig.14a). These levels are maintained during the entire time of representation of the given digit. The potential code for representation of binary numbers is used, for example, in the Soviet computer "Strela". In this computer,  $U_h \approx 200 - 300$  v,  $U_l \approx 50$  v.

In the dynamic method, the binary digits are represented by pulses of definite duration. Usually a one is represented by the presence of a pulse and a zero by its absence (Fig.14b). In some computers, one and zero are represented by pulses of different polarity (Fig.14c). In the latter case, the operation of the machine is more reliable, but its size is greater. The pulse code is used, for instance, in the Soviet computer "Ural".

Representation and transfer of binary numbers. To represent and transfer binary numbers in digital computers, the following codes are used: serial (time-pulse), parallel (space-position), and number-pulse. /66

In the serial code, a binary number is transferred along a single circuit in sequence in consecutive time positions, digit by digit. If the digits are represented by the static method, the voltage level is held constant on transfer of several similar digits in sequence (Fig.15).

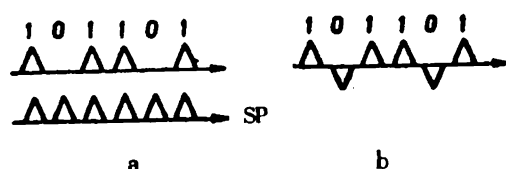


Fig.16 Transfer of a Number in Serial Code with the Digits Represented by the Dynamic Method

a - Ones are represented by the presence of pulses, zeros by their absence; b - Ones and zeros are represented by pulses of different polarity

In cases where the ones are represented by the presence of pulses and the zeros by their absence, the transfer of numbers by the serial code requires

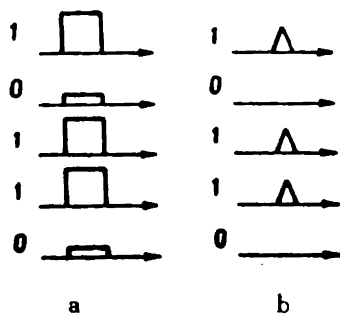


Fig.17 Transfer of Numbers in Parallel Code

a - Numbers represented by the static method; b - Numbers represented by the dynamic method

that their digits arrive at strictly determined instants of time. The time markers for the passage of each digit are formed by special synchronizing (sync) pulses SP (Fig.16a). If zero and one are represented by pulses of different polarity, there is no need for sync pulses (Fig.16b). The advantage of the serial code lies in the fact that only a single channel is needed to transfer numbers; however, such transfer does take a relatively long time. This time is determined by the number of digits contained in the number being trans-



ferred.

In the parallel code, all digits of the number are transferred simultaneously, each in its own channel (Fig.17). The advantage of the parallel code is that the time of transfer for the entire number is the same as the time of transfer of a single digit, but as many channels as there are digits in the number to be transmitted are required.

In the number-pulse code, a number is represented by a series of high-frequency pulses. A number of pulses in a series equals the value of the number; thus, if it is desired to transfer the number 310 (decimal), then the number of pulses is 310. This method is applicable to any number system. The pulses are counted by counters operating in the corresponding systems of notation. The use of the number-pulse code is convenient on input and output units of the "shaft-digitizer" and "digital-to-shaft" type, since the angle of rotation of a shaft can easily be represented by the number of pulses proportional to it. /67

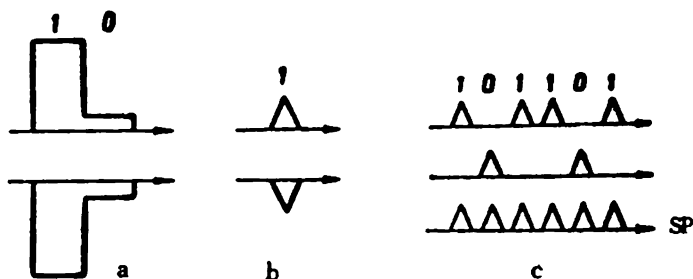


Fig.18 Representation of Numbers by the Paraphase Code  
a - Digits represented by the static method; b and c - Digits represented by the dynamic method

Depending on the code used, digital devices are classified into parallel-operation and serial-operation units.

The former have a higher speed than the latter, but also require more hardware.

In some units of digital computers so-called paraphase codes are used for improving the operational reliability and for supervisory purposes. Such units are used when the numbers are represented either by a parallel or a serial code. Twice the number of channels are required to transmit each digit of the number. The signals are fed into the first channel as in the parallel or serial code. On the second channel, voltages and pulses of opposite polarity to those on the first channel are transmitted (Fig.18a, b). In some units, zero is transmitted on the second channel instead of one, and one instead of zero (Fig.18c). The use of paraphase codes substantially increases the size and weight of the transmitting and receiving apparatus.

## ELEMENTS AND UNITS OF ELECTRONIC DIGITAL COMPUTERS

Section 16. Classification of Elements

The elements of electronic digital computers are mostly of the same type, which considerably enhances their technological effectiveness.

There are several basic characteristics by which digital computer elements can be classified.

According to purpose, the elements of computers are classified into memory, logic, and auxiliary (amplifying and shaping elements).

Memory elements serve to receive, store, and put out signals consisting of digital codes. The present forms of machine memory are rather complex devices consisting mainly of memory elements (ferrite cores with windings, elementary surface areas of magnetic drums for recording single symbols, etc.). The most widely used are flip-flops with two stable states. The transition from one state to another takes place under the action of signals arriving from without. Flip-flops are used for the reception storage and output of the binary digits or bits 0 and 1. Electron tubes, semiconductor diodes and triodes, ferrite cores with a rectangular hysteresis loop, parametrons, etc., are used as flip-flops. These elements are also used in combination to record binary digits, for example static and dynamic triggers, ferrite-diode and ferrite-transistor cells, etc.

Logic elements are used to control the operation of the memory elements and of individual assemblies, modules, and units of the computer, and also to build logic circuits for realizing logic functions.

The basic logic operators NOT, AND, OR are realized by the basic logic elements whose symbolic designations are given in Figs.8, 9, and 10.

The operation of logical negation is realized by inverters (NOT gates). When a signal of positive polarity is fed to the input of an inverter, a signal of negative polarity appears at the output, and vice versa. /69

To realize logical multiplication, coincidence circuits are used. In general, these are multipoles with  $n$  inputs and one output. A signal will appear at the output if and only if there are signals at all inputs simultaneously. In designing coincidence circuits with ferrite-diode or ferrite-transistor cells of certain types, the signals need not necessarily be fed simultaneously to the input in order to obtain a signal at the output.

To realize the OR operator (logical addition), collecting circuits, also called buffers, are used. A signal will appear at the output of a buffer if

there is a signal on at least one of its  $n$  inputs.

The amplifying and shaping elements serve to shape, amplify and reshape the pulse signals, and to restore the potential levels to their nominal value. These include amplifiers, pulse generators, pulse shapers, cathode follower, and other circuits.

Cathode followers are usually connected to the potential outputs of memory elements (mainly static triggers) so as to improve the noiseproof feature and ensure coordination with other elements. Pulse shapers are used to restore the pulses in amplitude, shape, and duration. Potential and pulse amplifiers are used to amplify the signals. Potential amplifiers are used to restore or increase the potentials transmitted on circuits of the computer. Pulse amplifiers are used to amplify current pulses.

Not all of these auxiliary elements need necessarily be used in a computer. In many cases, especially when improved circuits are used for the memory and logic elements, cathode followers and amplifiers become unnecessary.

According to character of the code signals at the inputs and outputs, the elements of a computer can be subdivided into potential, current, potential-current, and phase.

To represent binary digits in potential elements, potentials of different levels are used. Potential elements are usually made of semiconductor devices and electron tubes. They are characterized by small currents in the input and output circuits, resulting in a low power consumption. According to type of the potential circuits, triggers, gates, inverters, and the like can be built.

In current elements, binary digits are represented by currents of definite magnitude and direction. The shape and amplitude of current pulses in the input and output circuits must meet severe requirements. This is responsible for a number of peculiar features of such elements, by comparison with potential 70 elements. Current elements include ferrite-diode cells, ferrite-transistor cells, etc.

Both currents and potentials are used in voltage-current elements to represent 1 and 0. For example, in the electron-tube recording amplifiers used in magnetic-drum or magnetic-tape memory units, control at the input is accomplished by potential signals, while the output signal is a current in the secondary of a transformer connected to the plate circuit of an electron tube.

In phase elements, the code signals in the input and output circuits are represented by electric oscillations differing in phase by  $\pi$  radians. One value of the phase is made to correspond with the code "1", the other with "0". Phase elements include inductive and capacitative parametrons.

The code signal at the output of potential, current, and voltage-current elements may be either pulsed (pulses of current or voltage) or static. Accordingly, we distinguish elements with a pulsed output signal and elements with a static output signal. For example ferrite-diode cells and ferrite-transistor

cells are current elements with pulsed outputs. A static trigger using electron tubes or semiconductor triodes is a typical example of a potential element, with an output at which either a static signal or a pulsed signal can be formed.

A special group are the so-called dynamic elements, which in position 1 give a continuous pulse train at the output, while in position 0 there are no pulses at the output.

Elements with pulse or static outputs are in turn subdivided into elements with pulsed inputs, elements with static inputs, and elements with pulsed and static inputs.

According to principle of operation we distinguish elements of the accumulative type (pulse counter) and elements of combinatorial type (positional code).

The code signal at the output of an accumulating element, representing the result of performance of a certain operation, is formed only after a certain number of successive signals have arrived at its input, usually a single input. The output signal does not disappear after arrival of the last input signal. The accumulating element is reset to its initial state by a special erase signal. A trigger with a counter input is a typical accumulating element.

The code signal at the output of a combinatorial element is formed only for a certain definite combination of input signals applied simultaneously. It is characteristic of a combinatorial element that the signal at the output appears simultaneously with the application of the input signals and disappears simultaneously with the interruption of these signals (neglecting the time <sup>/71</sup> of transient processes). Combinatorial elements include AND, OR, NOT, etc. logic circuits. They also include logic circuits with ferrite cores, although the production of output signals in these circuits is nonsimultaneous with the application of input signals.

The general requirements for all computer elements are high reliability under real operating conditions, sufficiently high operating speed, and smallest possible size and cost. The elements of which the operative memory of the computer is composed must meet the extremely strict requirements. This is because any operation on the computer may involve repeated recourse to the operative memory, for counting the instructions and numbers taking part in the given operation and for entering the results of the operation. The time consumed by the elements for operation has a substantial effect on the speed of the operative memory and on the speed of the entire computer. Logical methods of accelerating counting should therefore be used, and time lags of element operation should be minimized in every way.

## Section 17. The Ferrite Core as a Bistable Element

One of the most widely used flip-flops is the toroidal magnetic core with a rectangular hysteresis loop. Its operation is based on the fact that ferromagnetic materials assume one of two stable states, corresponding to either

positive or negative remanent magnetic induction.

Ferromagnetic materials, or ferromagnetics, of which magnetic cores are manufactured, are characterized by excellent magnetic properties and by high resistivity (up to  $10^9$  ohm/cm). The magnetic permeability of a ferromagnetic substance is considerably higher than its magnetic constant  $\mu_0$ , and its value depends on the intensity of the external magnetic field applied to the substance, as well as on the preceding magnetic states. The principal ferromagnetic materials are iron, nickel, cobalt, and their alloys.

In the absence of an external magnetic field, a ferromagnetic substance has individual regions (domains) of random magnetization, each in a definite direction. The magnetic fields of such domains do not appear in the external space, since the various domains are magnetized in different directions.

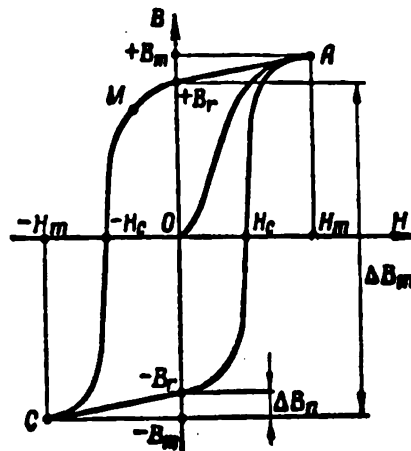


Fig.19 Hysteresis Loop of Ferromagnetic Materials

Consider the process of magnetization of a ferromagnetic substance. Let the substance initially be completely demagnetized, i.e., let its magnetic state be characterized by the point O (Fig.19). Let us now apply an external magnetic field to the substance. With increasing strength  $H$  of the external /72 magnetic field, the magnetic induction  $B$  increases, rapidly at first, since the elementary currents in the domains are so oriented that their magnetic fluxes add to the external flux. Thereafter, at greater values of the induction, the rate of its rise decreases; the magnetic state of the substance approaches saturation. Here, already almost all the elementary currents are so oriented that their magnetic fields coincide in direction with the external field. The curve OA is called the initial curve of magnetization. It characterizes the process of magnetization of a previously demagnetized ferromagnetic substance.

If the strength of the external field is brought up to the value  $H_m$ , then the magnetic induction equals  $B_m$  (point A). With decreasing strength of the

external field, the magnetic induction likewise decreases but the curve  $B = f(H)$  is then above the original magnetization curve. In a zero field, remanent magnetization of the substance and the related remanent magnetic induction  $+B_r$  still persist. This is explained by a certain degree of conservation of the ordered orientation of the elementary currents in the domains. The substance may remain for any desired time in the state characterized by remanent magnetic induction if no external magnetic field acts on it.

Reversing the direction of the external field, we begin at first to increase its strength. The magnetic induction will decrease, gradually at first, and then, at a certain field whose strength we denote by  $H_c$ , it will take a jump, causing reversal of the sign of induction. The field strength at which the magnetic induction changes sign is known as the coercive force  $H_c$ . With further increase of the field, the magnetic induction will increase smoothly to the value  $-B_r$ , which is reached at a field strength of  $-H_c$ . If the field strength is decreased, then the induction will also decrease and, at  $H = 0$ , the state of remanent magnetic induction  $-B_r$  will set in. This is the second stable state of a ferromagnetic material.

If a field of strength  $+H_c$  acts on a ferromagnetic substance of remanent induction  $-B_r$ , then the magnetic state of the substance will vary from the point  $-B_r$  to the point A. Thus, we obtain a closed hysteresis cycle. The <sup>173</sup> relation between induction and the strength of the external magnetic field is non-single-valued: The value of the induction for a prescribed field strength depends on the conditions under which the magnetization process has taken place. The closed curve, reflecting the character of the variation of the magnetic state of the substance under the action of an external field, is called a hysteresis loop.

The points of intersection of the hysteresis loop with the abscissa axis determine the coercive force ( $\pm H_c$ ) of a ferromagnetic material, while the points of intersection with the ordinate axis determine the remanent magnetic induction ( $\pm B_r$ ).

The coercive force determines the external field strength necessary for remagnetization or for magnetic polarity reversal of the substance. The smaller the coercive force, the less energy is required to flip the substance from one stable state into the other.

In general, different hysteresis loops will correspond to different values of the magnetic remagnetizing or switching field. If the fields are sufficiently great, however, they will differ less and less from each other and will tend to merge in some so-called limit hysteresis loop. The existence of a limit hysteresis loop makes  $B_r$  independent of the strength of the switching field, provided that the strength of this field is sufficiently high.

There are two groups of ferromagnetic materials used to make the flip-flop or bistable toroidal cores of digital computers.

The first group consists of strip or tape materials, used to fabricate strip toroids, consisting of several turns of permalloy or permivar strip

(based on a nickel-iron alloy). This strip, several microns thick, is wound on a ceramic spool, which protects its turns from deformation. The turns are welded together by spot welding. The toroid is then heat-treated in a desiccated atmosphere filled with hydrogen, under the simultaneous action of an external magnetic field.

Strip toroids have a relatively narrow hysteresis loop ( $H_c \approx 0.5$  oersted;  $H_r \approx 2$  oersted) high remanent and maximum induction ( $B_r \approx 13,000$  gauss,  $B_m \approx 15,000$  gauss), high eddy-current losses, and long remagnetization or switching time (from 6 to 20  $\mu$ sec) due to such losses, as well as good temperature stability of the operating characteristics.

The second group consists of molded or ferritic materials. Ferrites are complex metallic oxides. Their general formula is  $MOFe_2O_3$ , where M is any divalent element (Mn, Mg, Zn, Ni, Co, Cd, Cu). In the manufacture of ferritic toroidal cores, manganese-magnesium ferrites, made up according to the formula  $MgO \cdot MnO \cdot Fe_2O_3$  in the weight ratio 52 : 7 : 41 are most often used. /74

Ferrite toroids, like most other ferrite articles, are manufactured by the methods of powder metallurgy (cermet technology) which are as follows: The ferrite material is ground extremely fine in ball or vibration mills and then mixed with a plasticizer (an aqueous solution of polyvinyl alcohol or paraffin). Articles of the required shape are then pressed in metal dies from the mass so obtained and are finally heat-treated by roasting in chamber or tunnel furnaces at a temperature of the order of 1000 - 1400°C.

Ferrite cores have relatively wide hysteresis loops ( $H_c \approx 0.8 - 1.5$  oe;  $H_r \approx 4$  oe), low remanent induction and maximum induction ( $B_r \approx 2200 - 2500$  gauss;  $B_m \approx 2400 - 2700$  gauss), practically no eddy-current losses owing to their high electric resistivity and, therefore, short remagnetization times (0.5 - 1  $\mu$ sec).

Ferrite cores are cheaper and easier to work than strip cores but are inferior in temperature stability; with rising temperature, the shape of the hysteresis loop changes (becoming narrower), the remanent induction decreases, and the rectangularity ratio decreases. The greater the coercive force of a ferrite core, the less temperature-dependent are its characteristics. In circuits designed to operate in a wide temperature range or at elevated temperatures, it is therefore expedient to use cores of relatively high coercive force.

The power consumption for the control of the operation of the core is decreased by decreasing its dimensions.

Cores of strip materials are expediently used in cases where the operating conditions make it necessary to obtain greater signals at the output of the magnetic element or require high temperature stability.

In digital computer technology, ferrite cores are mostly used, and therefore in our further discussion of circuits with magnetic elements (ferrite-diode cells, ferrite-transistor cells, memory units using magnetic cores) we

will always mean ferrite cores.

If the positively magnetized state of a core ( $+B_r$ ) is taken as one, and the negatively magnetized state ( $-B_r$ ) is taken as zero, then the core can be considered a memory element with the two stable states 1 and 0. To obtain the state 1 it is sufficient to apply the positive magnetizing force  $+H_m$ , and to obtain the state 0 the negative magnetizing force  $-H_m$ . Conversely, sometimes the state  $+B_r$  of the core is taken as the state 0 and the state  $-B_r$  as the state 1. /75

The magnetic field that remagnetizes or switches the core from one stable state to the other is created by passing current pulses through its winding. Ferrite cores for logic circuits have at least three windings (Fig.20). The

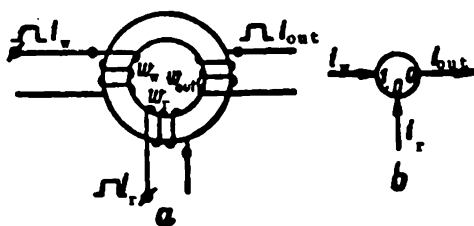


Fig.20 Ferrite Core with Windings  
a - Principal wiring diagram; b - Conditioned symbols

winding  $w_w$  is the write winding. When a write current pulse  $i_w$  of sufficient amplitude and duration is fed to this winding, the core is magnetized in the direction of positive induction and, after termination of the pulse, remains in the state  $+B_r$ , i.e., it writes 1. The winding  $w_r$  is called the read winding; when the read current pulse  $i_r$  is passed through it, the core is switched to the opposite magnetic state and, after termination of the pulse, remains in the state  $-B_r$ , corresponding to 0. So-called clock pulses, master pulses, or synchronizing pulses, spaced at a constant repetition rate determining the cycle of a digital computer, are often used as read pulses. In such cases the read winding is also called the clock or timing winding. Finally, the winding  $w_{out}$  is the output winding. The signals induced in it characterize the state of the core at the instant the read pulse  $i_r$  is fed to the winding  $w_r$ . In Fig.20 and the following figures the beginning of the windings is denoted by a dot.

In sketching complex circuits, symbols are generally used for the ferrite core. One of such symbols which is most widely used, is shown in Fig.20b where the circle corresponds to the ferrite torus. The arrows pointing toward the circle correspond to the write winding and the read winding, respectively, the arrow pointing away from the circle corresponds to the output winding. The figures inside the circle, relating to the output circuits (windings  $w_w$  and  $w_r$ ) indicate the state in which the core is placed on arrival of the pulses  $i_w$



and  $i_r$ . The figures relating to the output circuit show in what position the core must be set to obtain, at its output, the signal corresponding to the code "1". For example, if the code "1" is written by the current  $i_w$  in the core, then on arrival of the read pulse  $i_r$ , the core is switched into the state 0 and the code signal "1" will appear at its output.

If  $\tau$  which is the remagnetization time of the core, i.e., the time of switching from the state  $+B_r$  to the state  $-B_r$  or from the state  $-B_r$  to the state  $+B_r$ , is constant then the emf in the output winding will be proportional to the change in magnetic induction (Fig.19) after the time  $\tau$ : /76

$$e_c = -\frac{w_{out} S}{\tau} \Delta B_m = -\frac{w_{out} S}{\tau} (B_m + B_r), \quad (32)$$

where  $e_c$  is the emf representing the useful signal of the code "1", in volt;  
 $w_{out}$  is the number of turns in the output winding;  
 $S$  is the cross-sectional area of the core, in  $m^2$ ;  
 $\tau$  is the core remagnetization time, in  $\mu\text{sec}$ ;  
 $B_m, B_r$  are the saturation magnetic induction and the remanent magnetic induction in  $\text{weber}/m^2$ .

If the ferrite core was in the state 0 ( $-B_r$ ), then the pulse  $i_r$  will only slightly change its magnetic state. In this case, the consequence of the non-

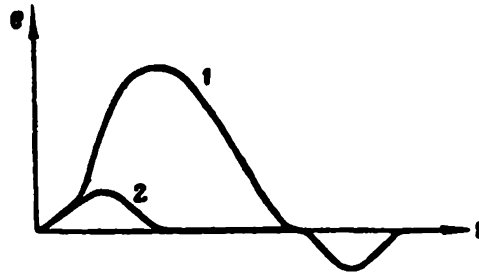


Fig.21 Graph of Variation of emf in the Output Winding of the Core in Reading 1 (Curve 1) and 0 (Curve 2)

ideal rectilinearity of the hysteresis loop during the rise and fall time of the pulse  $i_r$  is that small opposing changes by an amount  $\Delta B_n$  take place in the magnetic induction (Fig.19). Simultaneously with the change in magnetic induction, small noise pulses of opposite polarity appear in the output winding. Assuming the same duration of the leading and trailing edges of the read pulse, i.e.,  $\tau_{r \text{ l.e.}} = \tau_{r \text{ t.e.}}$ , then the emf of the noise  $e_n$  in the output winding is found from the formula

$$e_n = -\frac{w_{out} S}{\tau_{r \text{ l.e.}}} \Delta B_n = -\frac{w_{out} S}{\tau_{r \text{ t.e.}}} (B_m - B_r). \quad (33)$$

When cores are used in the various units of a digital computer, it is important to obtain as high a ratio of  $e_s/e_n$  as possible, i.e., the ratio of the magnitude of the useful signal arising in the output winding of the core in reading 1 to the magnitude of the noise arising in reading 0.

The character of the variation of the emf in the output winding of the core when it is completely remagnetized (reading 1) and likewise in the case of reading 0 is shown in Fig.21. The segment of the curve 1 above the abscissa characterizes the variation of the emf in the output winding when the magnetic induction varies from  $+B_r$  to the point C (Fig.19), i.e., on reading 1. The segment of the curve 1 below the abscissa characterizes the variation of the emf when the induction varies from the point C to  $-B_r$ . The positive half-wave of the curve 2 characterizes the variation of the emf on variation of the induction from  $-B_r$  to the point C (reading 0) and the negative half-wave on variation of the induction from point C to  $-B_r$  (restoration of the magnetic state of the core). /77

Ferrite cores are characterized by static and dynamic parameters. The static parameters determine the state of the core during the time of information storage. The dynamic parameters describe the process of switching or remagnetization of the core under the action of exciting fields.

The basic static parameters include the coercive force  $H_c$ , the remanent magnetic induction  $B_r$ , the maximum magnetic induction  $B_m$ , and the square ratio  $K_{sq}$ .

The square ratio is the ratio of the remanent magnetic induction to the maximum magnetic induction in determining the value of  $H_c$ :

$$K_{sq} = \frac{B_r}{B_m}.$$

For most ferrite cores the value of  $K_{sq}$  varies from 0.85 to 0.95. The greater  $K_{sq}$ , the smaller will be the variation of the magnetic induction on feeding the pulse  $i_r$  into the winding  $w_r$  of the core in the state  $-B_r$  (code "0"), and, consequently, the lower will be the value of the noise signals appearing in the output winding. In the ideal case, at  $K_{sq} = 1$ , there is no noise on reading 0 in the output winding.

The static parameters are determined from the limit static hysteresis loop, determined by the relation between the magnetic induction and the remagnetizing field strength:

$$B = f(H).$$

Commercial ferrite cores with rectangular hysteresis loop can be arbitrarily subdivided into two groups:

- a) cores used in memory units as elementary memory registers;
- b) cores used for building logic circuits.

In most cases, the ratios  $H_{e1} > H_{e2}$ ,  $B_{r1} < B_{r2}$ ,  $B_{s1} < B_{s2}$ ,  $K_{eq1} > K_{eq2}$  are valid; here,  $H_{e1}$ ,  $B_{r1}$ ,  $B_{s1}$ ,  $K_{eq1}$  are parameters of the cores of the first group, and  $H_{e2}$ ,  $B_{r2}$ ,  $B_{s2}$ ,  $K_{eq2}$  are parameters of the second group.

The dynamic parameters of cores are determined from the pulse characteristics, which are usually established on special test stands to record the results of periodic switching or remagnetization of the cores by a series of current pulses of known amplitude, polarity, and duration.

The most important pulse characteristics are:

78

a) the characteristic of switching time against applied field

$$\tau = f_1(H);$$

b) the characteristic of the dependence of switching speed on the applied field

$$\frac{1}{\tau} = f_2(H);$$

c) the characteristic of the dependence of the emf in the core windings on the applied field

$$e = f_3(H).$$

The pulse characteristics permit a determination of the necessary dynamic parameters such as switching time, power required for switching, magnitude of useful signal, noise level, switching factor, etc.

The switching time of a core with a rectangular hysteresis loop can be calculated from the formula

$$\tau = \frac{S_w}{H - H_0}, \quad (34)$$

where  $S_w$  is the switching factor of the core, or the switching constant;

$H$  is the switching field strength;

$H_0$  is the threshold field, or minimum magnetic field strength that will switch the core (usually  $H_e > H_0$ , since cores do not possess an ideal rectangularity of the hysteresis loop).

The quantities  $S_w$  and  $H_0$  are physical parameters of the core, and are determined by the composition of the core material, its structure, and the technology of core manufacture. The value of  $S_w$  can serve as a criterion of the core quality. The smaller  $S_w$ , the faster will the core be remagnetized or switched. In general, the values of  $S_w$  and  $H_0$  are not strictly constant for a given core and depend on the value of the switching field. However, if  $(2 - 3) H_e \leq H \leq (8 - 10) H_e$ , which is characteristic for cores in ferrite-diode cells and ferrite-transistor cells, the quantities  $S_w$  and  $H_0$  can be considered constants.

For a given ferrite core, the quantity  $S_w$  is determined from the formula

$$S_w = \tau \left( H - H_c \frac{2D}{D+d} \right) \quad (35)$$

where  $D$  is the outside diameter of the core;  
 $d$  is the inside diameter of the core.

79

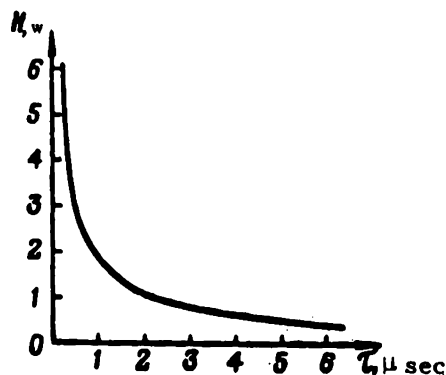


Fig.22 Graph of the Relation  $\tau = f_1(H)$

The relation between  $\tau$  and the switching field is of a form similar to that of the curve in Fig.22. The greater the value of  $H$ , the shorter will be

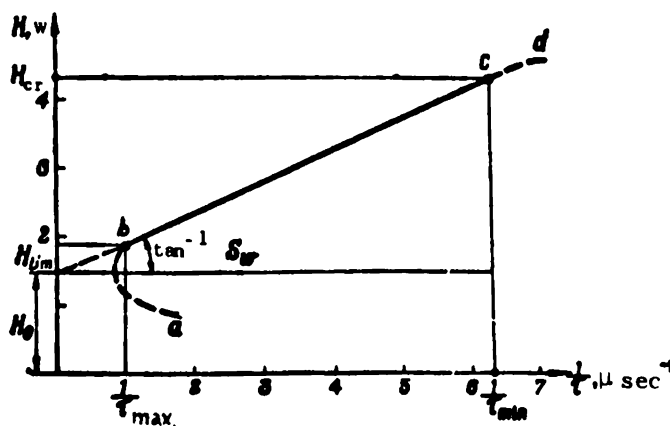


Fig.23 Graph of Relation  $\frac{1}{\tau} = f_2(H)$

the switching time.

In practice, the relation between the switching speed and the switching field is used more often. Figure 23 gives the general form of this relation. Until the switching of the core is accomplished on the limit hysteresis loop or,

In other words, on the limit cycle, the switching speed decreases with decreasing value of  $H$  [the segment c-b of the curve  $\frac{1}{\tau} = f_2(H)$ ]. When  $H$  decreases

below the limit  $H_{lim}$ , the core begins to switch on one of the partial cycles, and the switching speed increases (segment b-a of the curve). For  $H > H_{cr}$ , we observe a curving of the characteristic (the segment c-d of the curve) since the switching time becomes equal to the duration of the leading edge of the switching pulse ( $H_{cr}$  being the critical value of the field corresponding to the end of the linear segment of the curve). At the point of intersection of the prolongation of the linear segment of the curve with the  $H$  axis, we get the value of the threshold field  $H_0$ . The tangent of the angle of slope of the

linear segment of the curve  $\frac{1}{\tau} = f_2(H)$  to the abscissa gives the switching factor [eq.(34) is valid precisely for the linear segment of this curve].

Thus the characteristic  $\frac{1}{\tau} = f_2(H)$  makes it possible to determine the parameters  $S_w$ ,  $H_0$ ,  $H_{lim}$ ,  $H_{cr}$ ,  $\tau$ .

The most important quality factor of a ferrite core is the temperature dependence of its parameters. With rising temperature, the parameters  $B_s$ ,  $B_r$ ,  $H_s$ ,  $H_0$ ,  $S_w$ , and  $K_{sq}$  will decrease. The switching of a core under the action of a pulsed magnetic field takes place at higher speed with increasing temperature, so that the signal emf  $e_s$  will increase [see eq.(32)]. The noise emf  $e_n$  also increases, since the rectangularity of the hysteresis loop decreases [the value of  $\Delta B_s$  in eq.(33) increases].

A knowledge of the static and dynamic parameters of ferrite cores is necessary for the construction of circuits using these elements.

## Section 18. Basic Logic Elements using Ferrite-Diode Cells

Operating principle of ferrite-diode cells. In building logic circuits that realize both the simplest and composite logic functions, the ferrite cores with rectangular hysteresis loop discussed in the last Section are used in most cases in conjunction with semiconductor triodes (ferrite-diode cells) or triodes (ferrite-transistor cells). The memory element in such cells is a ferrite core, while the diodes and triodes perform auxiliary functions.

A ferrite-diode cell is a combination of a ferrite core with at least three windings and a semiconductor diode in a certain definite way. In complex circuits, ferrite-diode cells form either series circuits or parallel circuits or else branched circuits. It is convenient to base the discussion on the operating principle of a ferrite-diode cell which is an element in a series circuit composed of cells of the same type.

Figure 24 shows three ferrite-diode cells connected in series. Each cell contains a ferrite core (FC) with a write winding  $w_w$ , a read winding  $w_r$ , and an

output winding  $w_{out}$ , together with a diode  $D_1$ . The output winding of the preceding core is connected to the read winding of the next core across a coupling quadrupole containing in this case an LC circuit. The read windings of all cells are connected in series. Thus the reading ampere-turns produced by the 81 current  $i_r$  act simultaneously on the cores of all the cells. This circuit for the cells, when it is necessary to have read pulses (clock pulses) of only one series, is called a single-cycle circuit or a circuit with single-cycle feed.

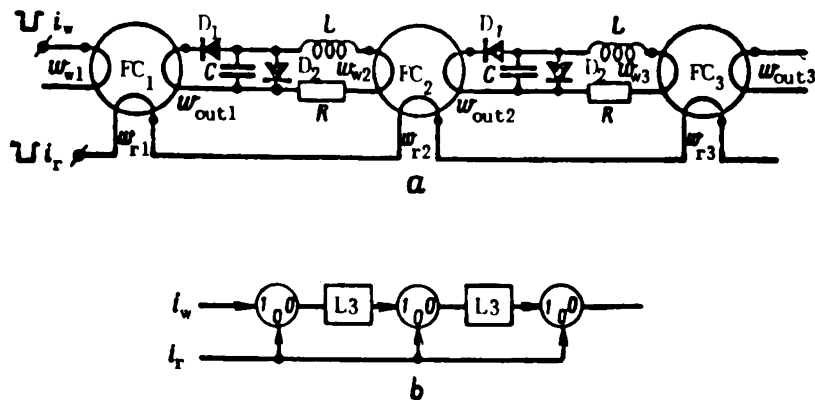


Fig.2.4 Series Connection of Ferrite-Diode Cells with Single-Cycle Feed:  
a - Schematic circuit diagram; b - Symbolic representation

In the circuit under discussion and also in all the following circuits in which ferrite cores with a rectangular hysteresis loop are used, we will assume that to the state 1 there corresponds the positive magnetization  $+B_r$  and to the state 0, the negative magnetization  $-B_r$ .

Let a write pulse  $i_w$  be fed to the read winding  $w_{r1}$  of the core  $FC_1$ . The core  $FC_1$  is switched to position 1, and an emf induced in its output winding is applied by the positive pole to the cathode of the diode  $D_1$ . The diode  $D_1$  is blocked, and no current flows in the connecting circuit between the cores  $FC_1$  and  $FC_2$ . On arrival of the read pulse  $i_r$ , the core  $FC_1$  is returned to the zero position, and the emf induced in its output winding is now applied by the negative pole to the cathode of the diode  $D_1$ . As a result, the diode  $D_1$  is opened, and in its connecting circuit a current begins to flow charging the capacitor  $C$ . Thus, on switching the core from position 1 to position 0, i.e., on reading off 1, the energy of the pulse  $i_r$  is transferred to charge the capacitor. The diode  $D_1$  plays the role of a release element, ensuring the transfer of energy in the forward direction (from left to right) only during the time of the read-out cycle.

The remagnetization of the core  $FC_2$  into position 1 is accomplished by the discharge current from the capacitor. The inductance  $L$  and the resistor  $R$  are

connected in the discharge circuit. Owing to the inductance, the current in /82 the discharge circuit does not reach a maximum immediately but only with a delay which should exceed the duration of the readout pulses. This delay is necessary in order for the discharge current to switch the next core only after the end of the readout pulse, which holds the core in the zero position. The transfer of energy along the circuit between the cores must be accomplished with minimum loss. These losses actually consist of the losses in the charging circuit (the voltage drop across the open diode  $D_1$  and in the discharge circuit of the capacitor). More energy can be accumulated in the capacitor than is necessary for switching the next core, and may lead to operational dropout owing to oscillatory processes in the discharge circuit. On account of the resistor  $R$ , these oscillatory processes are substantially damped.

The coupling quadrupole must ensure the transfer of energy with minimum losses only in the forward direction, and the transfer of energy from one core to another must be accomplished with minimum delay which, however, must exceed the duration of the readout pulse. The transfer of energy in the opposite direction from right to left, besides the expenditure of additional energy, may also interfere with the reliability of operation of the circuit, because of a parasitic partial switching of the cores in the zero position. In fact, on transmission of the code "1" from the core  $FC_2$  to the core  $FC_3$  which, like the core  $FC_1$ , is in the zero position, an emf induced in the winding  $w_{w2}$  under the action of the pulse  $i_r$  causes current to flow in the connecting circuit with the core  $FC_1$ . The core  $FC_1$  is partially switched, so that its remanent magnetic induction decreases. After the next regular readout pulse, which transmits the code "1" from the core  $FC_3$  to the next core, the magnetic state of the core  $FC_2$  varies somewhat more than that of the core  $FC_1$  in the previous pulse. The explanation is that the core  $FC_2$  is affected (tending to switch it to position 1) on the one hand by the increased noise pulse from the core  $FC_1$  when 0 is read out of it and, on the other hand, by the inverse noise pulse of the core  $FC_3$  when 1 is read out of it. The result is that, with each readout pulse, the noise signal increases on the capacitor of the cell preceding the one from which 1 was read out. In operating at elevated temperature or at a relatively small square ratio, the noise may be comparable to the useful signal, leading to an interference with proper operation of the unit consisting of series-connected ferrite-diode cells.

The transfer of energy in the opposite direction is minimized on account of the inductance  $L$  and the connection of the additional shunting diode  $D_2$ . During the time of action of the readout pulse, which forms the emf in the winding  $w_w$  of the core from which 1 is read, the current through the inductance cannot increase appreciably. This results in the circuit of the winding  $w_{out}$  /83 of the preceding core tapping only an insignificant part of the current flowing through the winding  $w_w$  of the core being switched by the pulse  $i_r$ .

A unit with series-connected ferrite-diode cells assembled on the basis of the circuit shown in Fig. 24 does not operate with sufficient stability at fluctuations in feed voltage and temperature over a wide range. When the feed voltage increases the power of the pulse  $i_r$  will also increase while, with rising temperature, the energy necessary to switch the core will decrease as a result of the decrease in remanent magnetic induction and coercive force. In

both cases, energy in excess of the energy required to switch the following core will accumulate on the capacitor. In that case, the discharge current from the capacitor reaches a maximum only after switching of the following core has been completed. The remainder of the energy is concentrated in the choke  $L$ . When the discharge current of the capacitor decreases, the voltage across the choke changes its polarity and tends to support the current. This reactive current can be closed through the coil  $w_{out}$  of the preceding core, thus causing its partial switching. Thanks to the connection of the diode  $D_2$ , the share of the reactive current of the choke closed across the winding  $w_{out}$  will decrease, which helps to improve the stability of operation of the circuit. At considerable fluctuations in feed voltage or temperature, however, this measure proves inadequate.

To ensure reliability of operation of the circuit (Fig. 24) and to satisfy the requirements to be met by coupling quadrupoles, it is advisable to replace the inductance by an electronic key in the connecting circuit, this key being open during the time of action of the readout pulse and closed during the discharge of the capacitor through the input circuit of the following core. Control by an electronic key may be accomplished in two ways (Bibl. 9): either by holding the key closed all the time, opening it during passage of the current  $i_p$  across the read winding or, holding the key open all the time, closing it after the end of the readout pulse for the time necessary for the capacitor to discharge.

Figure 25 shows a circuit of series-connected ferrite-diode cells with single-cycle feed and with a common electronic key for all coupling quadrupoles. The electronic key used is a semiconductor triode which, in the absence of a control voltage pulse  $u_{con}$ , remains in the open state on account of the negative voltage applied to its base through the resistor  $R_1$ .

The circuit operates in the following sequence: Let the core  $FC_1$  be in the position 1, and the remaining cores in the zero position. Simultaneously with the arrival of a readout pulse which switches the core  $FC_1$  into the zero <sup>(8)</sup> position, a positive control voltage is applied to the base of the triode across the diode  $D_3$ , thus blocking the triode. The emf  $e_{out}$  (Fig. 25b) arises in the winding  $w_{out}$  of the core  $FC_1$ , and the capacitor  $C$  is charged. Before the end of the control signal  $u_{con}$ , the voltage across the plates of the capacitor  $u_c$  remains practically constant, since the discharge current  $i_p$  is zero (the electrical circuit for the discharge current is open since the triode is blocked). The blocking signal  $u_{con}$  must have a duration not shorter than that of the readout pulse.

The core  $FC_2$  is switched from the zero position to the one position by the discharge current from the capacitor  $i_p$ , which begins to flow as soon as the pulse  $u_{con}$  ends. At the initial time, the current  $i_p$  is determined by the counter-emf induced in the winding  $w_{n2}$ . The discharge current increases sharply when the core  $FC_2$  is completely switched (in this case  $e_{n2} = 0$ ). The purpose of the diodes  $D_2$  is to prevent the charging of capacitors belonging to other cores during the transfer of energy from a given core. If, for example, energy is transferred (advance of the code "1") from  $FC_2$  to  $FC_3$ , then the diodes  $D_2$  will prevent charging of the capacitors  $C_1$  and  $C_3$ .



Synchronization of the pulses  $u_{con}$  and  $i_r$  is a necessary condition for reliable operation of the circuit. Under this condition, the circuit is practically insensitive to fluctuations in duration and repetition frequency of the

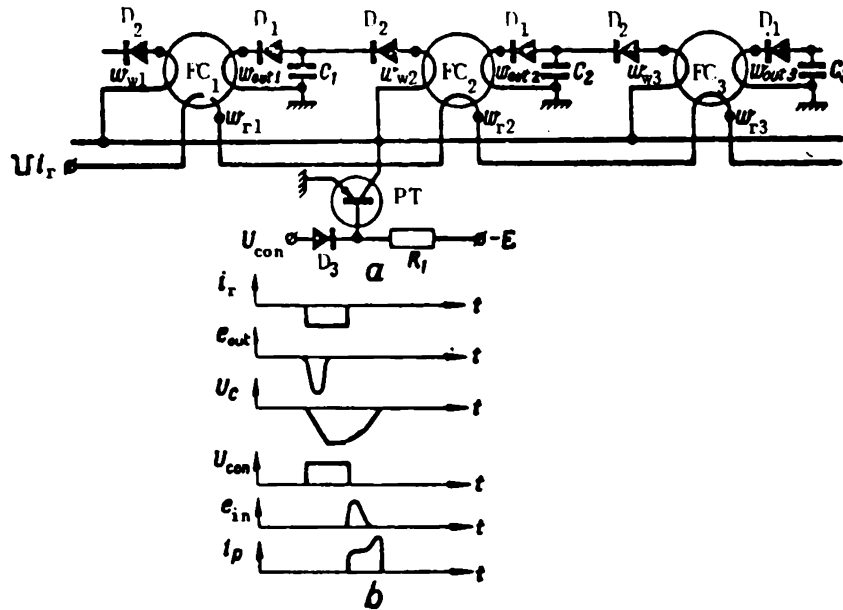


Fig.25 Single-Cycle Diagram for Series Connection of Ferrite-Diode Cells with a Common Control Key for all Quadrupoles of the Connection

a - Principal wiring diagram; b - Flow chart of operation

pulses  $i_r$ , since the discharge of the capacitor takes place at the end of the pulse  $i_r$ .

A disadvantage of this circuit is the need for constant consumption of energy in keeping the triode in the open state.

The reliability and speed of the advance of the code "1" along a chain of series-connected ferrite-diode cells increases when the most favorable circuits of the coupling quadrupoles are selected. The considerable inertia of the processes in the discharge circuit of the capacitor, however, limits the maximum rate of advance of the code "1" to a value of the order of 100 kc. This is a substantial shortcoming of single-cycle circuits using ferrite-diode cells. In addition, the design of such circuits involves certain difficulties, for example in obtaining the required duration of the readout pulses and in selecting the parameters of the coupling quadrupoles.

Computer technology tends more to favor two-cycle circuits or push-pull series connection of ferrite-diode cells (Fig.26). Here there are two trains of readout pulses ( $i_{r1}$  and  $i_{r2}$ ) shifted a half-period relative to each other. The pulses  $i_{r1}$  are fed through the bus I to the read windings of the odd-

numbered cores, while the pulses  $i_{r2}$  are fed through the bus II to the read windings of the even-numbered cores.

When the core  $FC_1$  is switched by the pulse  $i_{r1}$ , the code "1" is rewritten

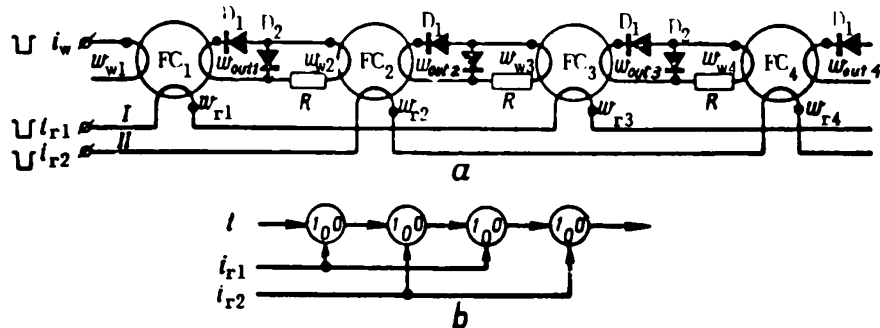


Fig.26 Series Connection of Ferrite-Diode Cells with Push-Pull Feed  
a - Principal wiring diagram; b - Symbolic representation

on the core  $FC_2$  after which, following a half readout-pulse period, the code "1" is rewritten by the pulse  $i_{r2}$  from  $FC_2$  to  $FC_3$ . The code "1" is further advanced by the pulse  $i_{r1}$ , etc. If the odd-numbered ferrite-diode cells are considered the principal cells, then the even-numbered auxiliary cells play the role of the coupling quadrupoles in single-cycle circuits.

The two-cycle circuit for connection of ferrite-diode cells differs little in economy from the single-cycle circuit. This is due to the fact that two sequences of readout pulses are required instead of one and that there is an unproductive consumption of energy transferred in the opposite direction (from right to left). If, for example, 1 is read out from the core  $FC_2$ , then part of the energy is lost in the circuit connecting it with the core  $FC_1$ , since the polarity of the emf induced in the winding  $w_{w2}$  is such that the diode  $D_1$  does not prevent passage of the current. A considerable decrease in the energy transfer in the opposite direction is obtained by a correct selection of the turn ratio in the windings  $w_{out}$  and  $w_w$  [usually,  $w_{out} = (2 - 3) w_w$ ], and by using the shunting diode  $D_2$  and the resistor  $R$ . But this does not completely eliminate the flowback of energy; in addition, up to 25% of the energy of the read pulses is lost across the resistor  $R$ .

The two-cycle circuit or push-pull connection of ferrite-diode cells with controllable key triodes in the connection circuit between the cores is preferable (Fig.27); this eliminates the resistors and diodes shunting the flowback of energy. The key triodes  $PT_1$  and  $PT_2$  open alternately on arrival, across the buses I and II, of the readout pulses  $i_{r1}$  and  $i_{r2}$ , thereby closing the circuit of the output winding of the preceding core from the input circuit of the following core. In its other details, the circuit operates like that of Fig.26.

Its advantage consists in the fact that it completely prevents transfer of energy in the opposite direction.

The speed of push-pull circuits with ferrite-diode cells is no worse than that of single-cycle circuits and, in addition, such circuits are noncritical

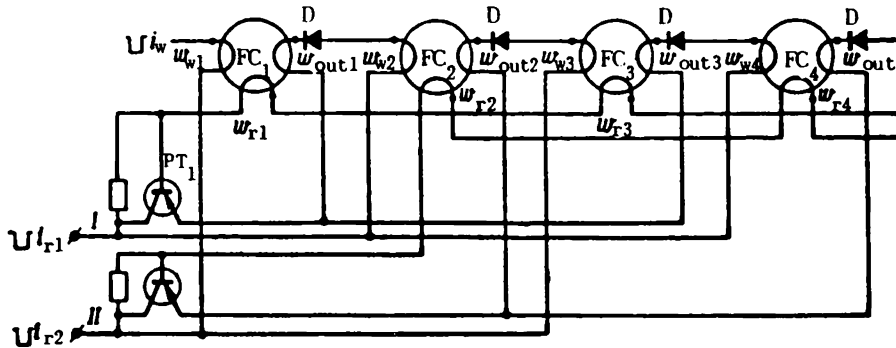


Fig.27 Push-Pull Circuit for Connection of Ferrite-Diode Cells to Controlled Key Triodes

to the duration of the readout pulses.

Three-cycle ferrite-diode circuits are also in use. These require three sequences of readout pulses shifted a third of a period relative to each other. /87

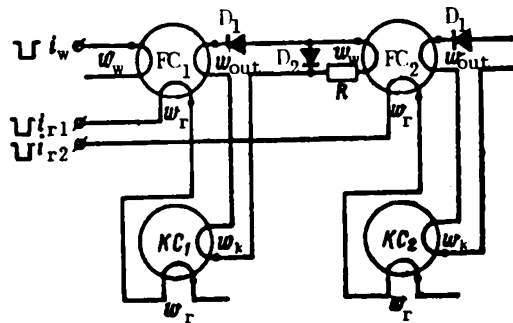


Fig.28 Push-Pull Circuit for Connection of Ferrite-Diode Cells to Compensating Cores

In designing circuits that use ferrite-diode cells it is important to decrease the noise caused by the nonrectangularity of the hysteresis loop of the ferrite cores. The most effective method of decreasing this noise, especially in operation over a rather wide temperature range, lies in the use of auxiliary compensating cores. Figure 28 shows a push-pull circuit of connecting ferrite-diode cells in which the number of compensating cores KC equals the number of principal cores. The output winding  $w_{out}$  of a main core and the compensating

winding  $w_k$  of the core KC are connected in push-pull, while their read windings are connected in the same direction. The number of turns is taken equal, i.e.,  $w_r = w_r^1$ ,  $w_{out} = w_k$ . With such a connection, the compensating core is always in the zero position. If the main and compensating cores have hysteresis loops of about equal square ratios, which is obtainable by proper selection of the cores, then the noise voltage in the winding of the main core arising during the read-out of 0 from it, will be almost completely compensated by the emf of opposite polarity induced in the winding  $w_k$ . Obviously, the use of a compensating core also involves a certain decrease in the signal of the code "1" read from the main core. It does, however, give an output signal-to-noise ratio of the order of 20 - 40.

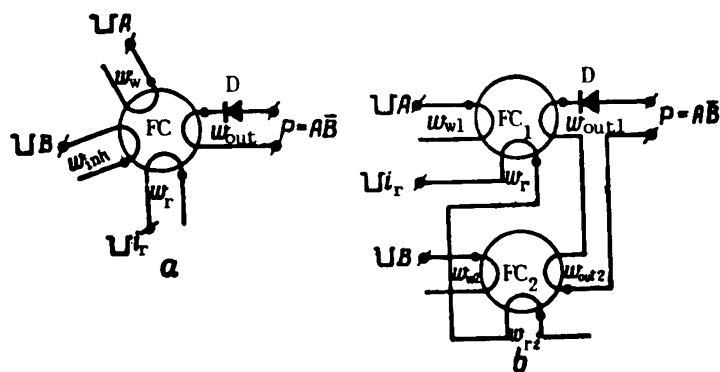


Fig.29 Inhibit Circuits

a - Circuit based on compensation of magnetic flux; b - Circuit based on compensation of emf in the output winding

The principal advantages of circuits with ferrite-diode cells are relative simplicity and low cost. Their disadvantages include the following:

- a) the need for a generator of sufficiently powerful readout pulses  $i_r$  (the ferrite-diode cell being the passive element, and the source of energy being the generator of the currents  $i_r$ );
- b) energy losses in the coupling quadrupoles;
- c) existence of flowback of energy, which can be decreased or eliminated only by shunting diodes and resistors, electronic keys, and the like;
- d) low speed: the read pulse repetition rate in practical circuits does not exceed 100 - 150 kc;
- e) the need for controlling the noise due to nonrectangularity of the hysteresis of the cores. /88

In spite of this, in computer technology ferrite-diode cells are used for building registers, frequency dividers, switching circuits, logic circuits, etc.

These disadvantages of circuits with ferrite-diode cells have given an incentive for developing the use of active connections between the cores, so as to ensure replacement of the energy lost on core switching. The use of junction transistors as the active elements has proved to be most expedient.

Let us consider the operation of the principal logic elements constructed of ferrite-diode cells.

Inhibit circuits. In building inhibiting circuits with ferrite-diode cells, there are two principal methods of compensating the signals: compensation of the magnetic flux of the core in the input circuit and compensation of the voltage in the output circuit of the cell.

If the former method is used, the ordinary ferrite-diode cell is provided with a fourth winding called the inhibit winding  $w_{inh}$  (Fig.29a), which is connected in push-pull with the write winding. Consequently, when current pulses (signals A and B) arrive simultaneously in the write and inhibit windings, the resultant magnetic fluxes compensate each other and the code "1" is not written in the core. The readout pulse does not change the magnetic state of the core, so that there is no signal at the output ( $P = 0$ ). The code signal "1" at the output will appear only for one combination of input signals  $A = 1, B = 0$ . In other words, the circuit realizes the logic function

$$P = AB.$$

For stable operation of the circuit, the input signals A and B must agree exactly in time.

/89

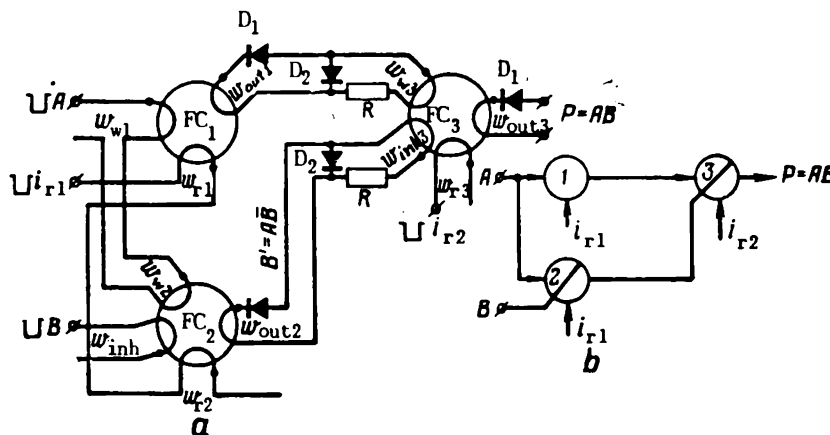


Fig.30 Coincidence Circuit with Ferrite-Diode Cells  
a - Schematic circuit diagram; b - Symbolic representation

When the latter method is used, the ordinary ferrite-diode cell is connected to still another core with three windings (Fig.29b). The windings  $w_{out1}$  and  $w_{out2}$  are connected in push-pull. If the code signals 1 and 0 are fed only to the input A, the ferrite core  $FC_2$  will act as a compensating core:

during the readout of 0 by the pulse  $i_r$ , the noise at the output will be compensated, while during the readout of 1, the amplitude of the code signal at the output is somewhat decreased on account of the emf of opposite polarity induced in the winding  $w_{out 2}$ . If the code signals are fed to both inputs, the cores  $FC_1$  and  $FC_2$  are switched to the position 1 and then reset by the pulse  $i_r$  to the position 0. The emf arising in the output windings are mutually compensated, so that we get  $P = 0$  at the output. In contrast to the preceding circuit, exact coincidence of the signals A and B in time is not required. All that is necessary is that the two signals are given before arrival of the regular readout pulse. It is, however, necessary to have reliable compensation in the output circuit, which is attained by proper selection of the ferrite cores.

Coincidence circuit. Realization of the logic function  $P = AB$  is accomplished by the circuit presented in Fig.30. The circuit is composed of one conventional ferrite-diode cell and two inhibit cells. The operation of these cells is based on compensation of the magnetic flux in the input circuit. The information is read out by read pulses of two series  $i_{r1}$  and  $i_{r2}$  which are a half-period out of phase with each other.

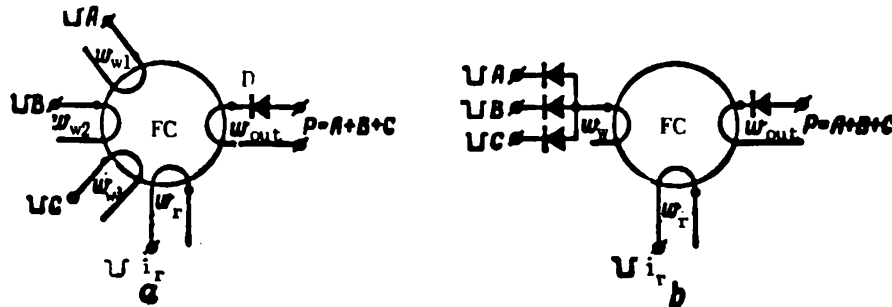


Fig.31 Common Collector Circuits

a - With separate read windings; b - With inputs bypassed by diodes

Let the signal code "1" be applied only to the input A of a coincidence circuit. Then both cores of the first stage  $FC_1$  and  $FC_2$  will be set in position 1. They will be reset by the pulse  $i_{r2}$  to the zero position; in this case, in the read winding  $w_{r3}$  and the inhibit winding  $w_{inh3}$  of the core  $FC_3$ , there will be currents induced by the mutually compensated magnetic fluxes. As a result, the core  $FC_3$  will remain in the zero position and, on arrival of the pulse  $i_{r2}$ , there will be no code signal 1 at the output of the circuit ( $P = 0$ ).

If the code signals 1 are applied to both inputs ( $A = 1, B = 1$ ), then only the core  $FC_1$  will be set in position 1. The core  $FC_2$  remains in the position 0, since the actions of the currents in its read and inhibit windings are mutually opposite. The code "1" is rewritten by the pulse  $i_{r1}$  from the core  $FC_1$  onto the core  $FC_3$  and is then transferred by the pulse  $i_{r2}$  to the output of the circuit ( $P = 1$ ).

Thus, this circuit realizes the operator AND, which is also confirmed by

the logic functions. Here, the cell 1 ensures transformation of the signal A, while the inhibit cell 2 ensures realization of the relation

$$B' = \overline{AB},$$

where B' is the signal at the output of cell 2.

The inhibit cell 3 realizes the relation

$$P = A\overline{B} = A\overline{AB} = A(\overline{A} + \overline{B}) = \overline{A}B,$$

where P is the signal at the output of cell 3 or at the output of the entire coincidence circuit.

Common collector circuit. The logic function  $P = A + B + C$  is simplest to realize. For this purpose, one uses common collector circuits either with separate read windings or with inputs bypassed by means of diodes (Fig.31). In both cases, the ferrite core is switched to position 1 on arrival of even a single input code signal 1 and is then reset by the pulse  $i_r$  to the position 0. In this case, the code signal 1 will appear at the output of the circuit.

## Section 19. Ferrite-Transistor Cells

/91

Operating principle of ferrite-transistor cells. A number of the disadvantages of ferrite-diode cells are eliminated by the use of ferrite-transistor cells which, to a certain extent, are a development from ferrite-diode cells. The passive element - the diode - is here replaced by an active element - a transistor - which yielded a substantial improvement in the characteristics of the cell.

The following types of ferrite-transistor cells are used in digital computers: simple cell, inhibit cell, field-magnetization cell and, less often, cell with self-readout (with self-excitation). The principal elements of a cell of any of these types are a transformer with a ferrite core having a rectangular hysteresis loop and the semiconductor triode. In all cases the memory element is a core, while the triode serves to amplify the signals and separate the circuits, preventing backflow of information.

The core of a simple ferrite-transistor cell (Fig.32) usually has four windings: the write winding  $w_w$ , the readout winding  $w_r$ , the base winding  $w_b$ , and the collector (output winding  $w_k$ ). The triode as a rule is connected by a circuit with grounded emitter, since such a circuit has the greatest power amplification and also has the necessary number of turns in the base and collector windings, making for fewer connections for the triode than in other circuits (with grounded collector or with grounded base). The base winding is connected to the input circuit of the triode, while the collector winding is connected to the load. The number of write and read windings to which exciting current pulses are fed may, in the general case, be greater. /92

The complete operating cycle of a ferrite-transistor cell consists of two cycles. We will consider that, in the initial position, the code "0" is stored in the cores of the cell, i.e., that the magnetic state of the core is characterized by the negative remanent magnetic induction  $-B_r$ . The triode is in

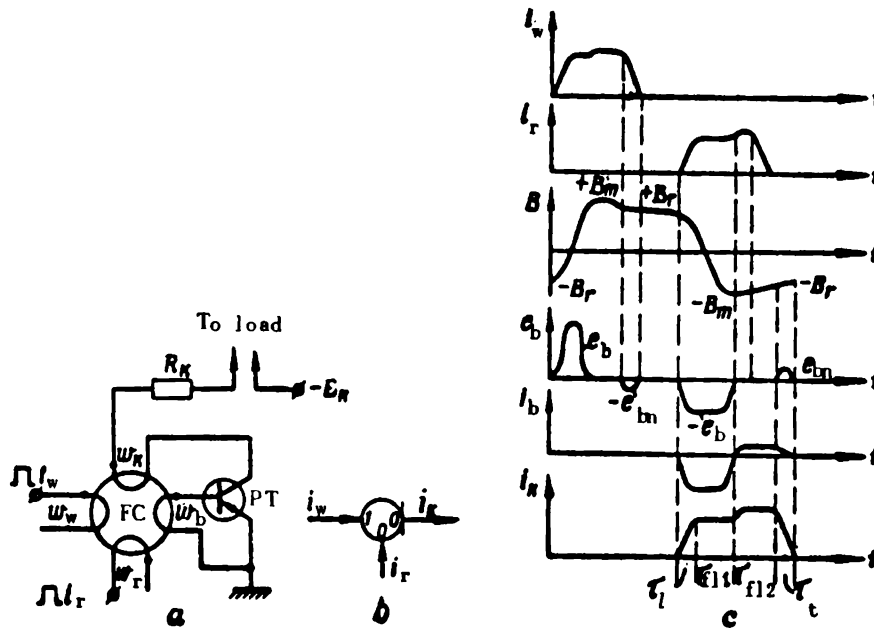


Fig.32 Ferrite-Transistor Cell  
a - Principal wiring diagram; b - Symbolic representation;  
c - Flow chart of operation

the blocked position, and a very small uncontrolled collector current flows in its collector circuit. During the first cycle, 1 is written; a write pulse  $i_w$  is fed to the write winding  $w_w$ , which switches the core from the position  $-B_r$  to the position  $+B_r$  (Fig.32c). The resultant emf  $e_b$  induced in the base winding is applied by the positive pole to the base of the triode, causing the triode to remain in the blocked state (in the circuit under discussion and in all following ones, unless specifically stated otherwise, we have in mind the use of type p-n-p semiconductor triodes). The core remains in the state  $+B_r$  until incipient decay of the write current pulse. During the period of the trailing end of the pulse  $i_w$  the magnetic state of the core changes from  $+B_r$  to  $+B_r$ . The noise emf  $-e_{bn}$  induced in the base winding is of a polarity opposite to that of the emf  $e_b$  induced during the writing of 1, and its duration equals the duration of the trailing end of the write pulse  $i_w$ . The emf  $e_{bn}$ , since it is applied by the negative pole to the base of the triode, tends to open the triode. However, because of the low amplitude and short duration of that pulse, the triode either remains in the closed position or is opened for only a short time, and then a noise current pulse arises in the collector circuit. At the end of the pulse  $i_w$ , the core remains in the position  $+B_r$ , corresponding to the code "1". In this position, it may remain indefinitely.

In the second cycle of operation of the ferrite-transistor cell, the



code "1" is out. For this purpose, the readout current pulse  $i_r$  is fed to the read winding  $w_r$ . The ampere-turns formed by the current  $i_r$  need be sufficient only to bring the core from the state  $-B_r$  to the state characterized by the point M of the hysteresis loop (Fig.19). During the switching, an emf  $-e_b$  is induced in the base winding. As a result, the triode is opened and the base current  $i_b$ , counteracting the switching, flows in the base-emitter circuit. The collector current  $i_c$  of the open triode, flowing through the winding  $w_k$ , which is a positive feedback winding, forms an additional readout field, accelerating the switching of the core. The switching process increases like an avalanche, and the triode enters the condition of saturation. After the core has reached the saturation  $-B_s$ , the emf in the base winding becomes equal to zero and the current in the base winding stops. However, even after complete switching of the core, the collector current  $i_c$  continues to flow for a time 198 which is determined by the resorption of the non-base carriers of the base current. Thus, the duration of the pulse  $i_c$  is determined by the choice of the type of triode and is composed of the core switching time and the triode cut-off time (the time of resorption of the non-base carriers of the base current). After termination of the pulses  $i_r$  and  $i_c$  the core passes into the state  $-B_r$  and, during transition from the state  $-B_s$  to the state  $-B_r$ , the emf  $e_b$ , which promotes blocking of the triode is induced in the winding  $w_b$ . The core remains in the state  $-B_r$  until arrival of the next regular write pulse.

If the read current pulse  $i_r$  (readout of code "0") is fed to the read winding  $w_r$  of the core in the zero position, then the magnetic state of the core is only slightly changed. On rise of the current  $i_r$  the core passes from the state  $-B_r$  to the state  $-B_s$ . In this case, the noise emf  $-e_b$ , arising in the winding  $w_b$  has the same effect on the triode as the noise emf arising during decay of the current  $i_w$  in the write pulse 1. After the end of the readout pulse, the core is returned to the position  $-B_r$ .

If the squareness factor of the hysteresis loop of the core is close to unity, then during readout of 0 there will be no collector current across the load since the triode remains in the blocked position. On the readout 1, on the other hand, a rather powerful current pulse  $i_c$  will flow across the load connected to the collector circuit of the triode.

The semiconductor triode in a ferrite-transistor cell operates under the saturation regime, performing the functions of a key. In this regime, the voltage of the feed source is most effectively utilized since the voltage drop across the triode in this case is only a fraction of a volt. Another advantage of the saturation regime is that the value of the collector current is practically independent of the triode parameters but is determined by the resistance of the load and the feed voltage. This permits building a ferrite-transistor cell that operates stably over a wide temperature range. Moreover, in the saturation regime the power dissipated in the transistors is considerably less than when the triode operates in the active domain. In circuits with ferrite-transistor cells, the regime of the triode in the active domain is rarely utilized, mainly on account of the strong dependence of the amplitude of the collector current pulse on the triode parameters.

The saturation regime is characterized by a constant value of the collector

current at the triode during resorption of the excess concentration of the non-base carriers of the base current. In a flow chart of the operation of a ferrite-transistor cell (Fig.32c) this is represented by the flat peak of the current  $i_k$  in the period  $\tau_{f1,2}$ . The current pulse  $i_k$  in the time interval  $\tau_{f1,1}$  has still another flat peak whose presence is explained as follows: The load in the collector circuit of a ferrite-transistor cell is usually represented /94 by the ferrite cores of other cells where the voltage drop across the windings varies with time. The switching of the cores of the load cells takes place in the time interval  $\tau_L + \tau_{f1,1}$  where  $\tau_L$  is the duration of the leading edge of the collector current pulse. In this case, the inductive resistance of the load rises and the magnitude of the current  $i_k$  in this interval will be less than in the interval  $\tau_{f1,2}$  at the beginning of which the process of remagnetization comes to an end.

The principal disadvantage of the saturation regime consists in the increased duration of the collector current pulse by the out-of-saturation time of the triode, which depends on the triode parameters. This explains the relatively low speed of units using ferrite-transistor cells. The maximum working pulse repetition rate, i.e., the repetition rate of the code pulses in the write winding or of the read pulses in the winding  $w_r$  is 100 - 200 kc.

The fact that the duration of the output pulse  $i_k$  of a cell whose triode is operating in the saturation regime exceeds the switching time of a core by the resorption time of the non-base carriers of the base current and the time of decay of the collector current is of primary significance in the construction of complex logic circuits with ferrite-transistor cells. In such circuits it often becomes necessary for the core of a ferrite-transistor cell to be remagnetized faster than the cores connected to the load circuit of that cell.

To simplify sketching of the circuits, a symbolic representation for the ferrite-transistor cell is introduced (Fig.32b). The ferrite core of the cell is denoted by a circle, its windings by lines with arrows, and the triode by a short line directed along a tangent to the circle. Hereafter, in considering wiring diagrams of ferrite-transistor cells of various types, we will assume that current pulses of positive polarity are fed to the separate terminals of the windings, whose beginning is marked by a dot.

The basic advantages of the simple ferrite-transistor cell are as follows:

a) relatively low requirements to be met by the ferrite cores and semiconductor triodes: a high uncontrolled collector current is admissible, and the permissible scatter of values of the amplification coefficients and the limiting frequency of the triode is greater than in purely semiconductor circuits;

b) low power consumption, since power is used only at the instant of switching the core (the triode is blocked the entire rest of the time). No high-power generators are required to form the readout pulses, since the required magnetic field in the readout of 1 is established on account of the positive feedback. In many cases, however, this last statement does not apply to the building of logic circuits with ferrite-transistor cells, since they involve the necessity for sufficiently powerful readout pulses, capable of /95

switching the cores without participation of positive feedback;

c) in circuits with ferrite-transistor cells, backflow of information is practically absent, which is explained by the unidirectional character of the amplifier (triode);

d) the load of a ferrite-transistor cell may consist of several such cells, thus facilitating construction of the branching circuits;

e) a ferrite-transistor cell can be used as a pulse former for pulses of definite amplitude and duration.

As already noted, during the write cycle on decay of the current  $i_w$  from its maximum value to zero and also during the read cycle on rise of the current  $i_r$  from zero to its maximum value, the triode of a ferrite-transistor cell may open briefly, accompanied by the appearance of a noise emf. At a sufficiently great squareness factor of the hysteresis loop of the core and a sharp trailing edge of the write current pulse  $i_w$ , the noise emf may reach a considerable value at which, because of the regenerative process, complete opening of the triode during the write pulse is possible. The probability of this event increases with the number of turns in the base winding and with the use of triodes having a high current amplification factor. Under these conditions, the cell may pass into the regime of self-readout (self-excitation), i.e., of operation on the trailing edge of the current pulse  $i_w$ . The phenomenon of premature output of information (readout during the write cycle) is in most cases undesirable. Spurious operation of the cell is prevented by connecting a source of constant positive bias in the triode base circuit. The voltage of this bias source will compensate the noise emf in both the write cycle and the readout cycle. The stability of operation of the cell can also be improved by introducing into the emitter circuit of the triode a small resistor (of the order of several ohms) so as to decrease the positive feedback in the cell.

When a positive bias source is connected to the base circuit, the time of removal of the triode from the saturation regime is shortened, thus increasing the speed of the cell.

These measures for improving the stability of operation of the cell, besides requiring additional energy consumption, are not sufficiently effective at large temperature fluctuations. Under such conditions, it is more effective to connect additional compensating cores in the circuit of the cell.

Figure 33 shows a circuit for a ferrite-transistor cell with one triode and three cores: an operating core (OC), a compensating write core (CWC), and a compensating readout core (CRC). The write windings of the cores OC and CWC are connected in push-pull, exactly as the read windings of the cores OC and CRC are connected. With such a connection, the compensating cores can be only in one of two stable states: the core CWC in the zero state ( $-B_r$ ), and the core CRC in the one state ( $+B_r$ ).

In writing 1 into the operating core, when the noise emf  $e_{b_n}$  appears in the base winding during the decay of the current  $i_w$ , an emf  $e_w$  is also induced in the base winding of the core CWC, but this emf has a sign opposite to that

of  $e_{bn}$ . If the squareness factors of the hysteresis loops of the cores OC and CWC are the same, then these emf compensate each other, so that the total emf across the base of the triode in the ideal case will be zero and there will

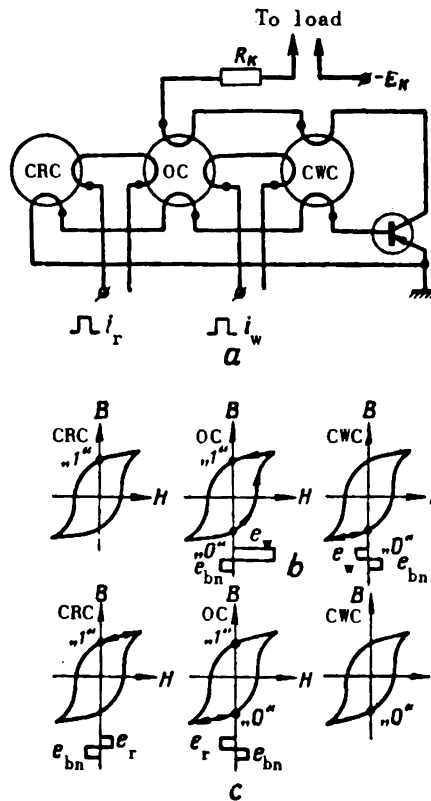


Fig.33 Ferrite-Transistor Cell with Compensating Cores  
a - Principal wiring diagram; b - Compensation of noise in writing; c - Compensation of noise in readout

be no noise at the output. In exactly the same way, in the readout of 0 from the operating core, the noise emf arising in its base winding (during rise of the current  $i_r$ ) is compensated by the emf  $e_r$  induced in the base winding of the CRC core.

The use of compensating cores complicates the ferrite-transistor cell. In matching the elements of the cell attention must be paid to the selection of cores with the same parameters. This complication can be tolerated if the unit using such cells must operate over a wide temperature range.

Ferrite-transistor inhibit cell (Fig.34). This cell differs from a simple cell by the presence of the inhibit winding  $w_{iab}$  on the core. This winding is connected in push-pull to the write winding, if the writing 1 is to be inhibited, or in push-pull to the read winding, if the readout of 1 is to be inhibited.

If the windings  $w_w$  and  $w_{inh}$  are connected in push-pull, then feeding the current pulse  $i_{inh}$  to the winding  $w_{inh}$  simultaneously with the pulse  $i_w$  will not change the magnetic state of the core, i.e., the writing of 1 is inhibited.

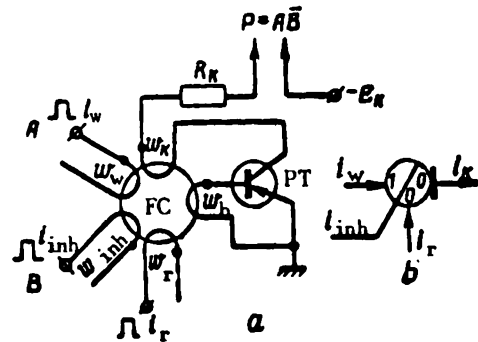


Fig.34 Ferrite-Transistor Inhibit Cell  
a - Principal wiring diagram; b - Symbolic representation

The reliability of execution of the operation of inhibition is increased if 197 the following condition is satisfied:

$$\tau_{inh} > \tau_w$$

$$i_{inh} w_{inh} \geq i_w w_w$$

where  $\tau_{inh}$ ,  $\tau_w$  are the respective durations of the inhibit pulse and the write pulse;

$w_{inh}$ ,  $w_w$  are the respective numbers of turns in the inhibit and write windings.

If the signals fed to the windings  $w_w$  and  $w_{inh}$  are denoted respectively by A and B, then the logic relation realized by the inhibit cell is of the form

$$P = AB,$$

where P is the signal at the output of the circuit.

The disadvantages of the inhibit cell are as follows:

- a) Two different types of shaping cells are required to shape the pulses  $i_{inh}$  and  $i_w$ , in view of the fact that  $\tau_{inh}$  and  $\tau_w$  are not the same.
- b) Strict synchronization of the pulses  $i_{inh}$  and  $i_w$  is necessary.

Ferrite-transistor cell with field magnetization (Fig.35). In this cell, the core has a field magnetization winding  $w_f$  instead of the write winding.

Through the winding  $w_f$ , flows a field magnetization direct current  $i_f$ , which maintains the core in the state of saturation corresponding to the code "1" (the point  $M_1$  of the hysteresis loop in Fig.35c).

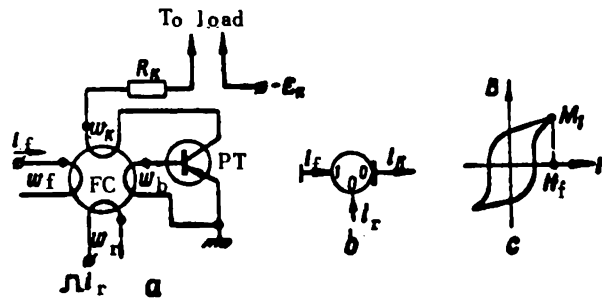


Fig.35 Ferrite-Transistor Cell with Field Magnetization on the Writing End  
a - Principal wiring diagram; b - Symbolic representation;  
c - Position of the working point on the hysteresis loop

The readout of 1 is accomplished by the pulse  $i_r$  and the information is regenerated by the current  $i_f$ . The readout ampere-turns are selected on the basis of the condition /98

$$i_r w_r \geq i_f w_f + H_c + i_b w_b$$

In such a cell the field magnetization is on the writing end. It is possible to build a circuit for a cell with field magnetization on the readout

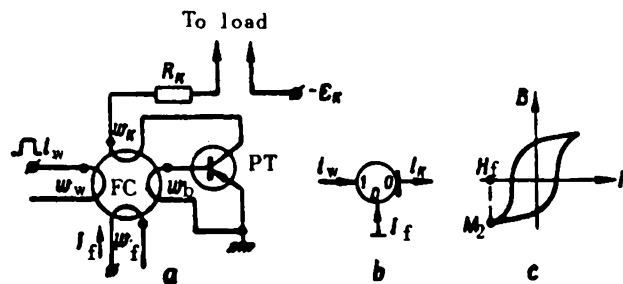


Fig.36 Ferrite-Transistor Cell with Field Magnetization on the Readout End  
a - Principal wiring diagram; b - Symbolic representation;  
c - Position of working point on the hysteresis loop

side (Fig.36), in which the core has a write winding  $w_w$ , a field magnetization winding  $w_f$ , a base winding  $w_b$ , and a collector winding  $w_k$ . The windings  $w_w$

and  $w_r$  are connected as random windings. The direct magnetization current  $I_r$  flowing through the winding  $w_r$  keeps the core in the domain of saturation corresponding to the point  $M_2$  of the hysteresis loop (Fig.36c). Consequently, the code "0" is at all times stored in the core. When the write current pulse  $i_w$  is fed, the core is switched to the state  $+B_1$ , and is then reset by the magnetization field to its initial position, while the code signal "1" appears at the output.

The regeneration of information after readout, when the action of the input signal has stopped, without a special pulse, is in many cases a valuable property of field-magnetized ferrite-transistor cells. In addition, such a cell is distinguished by a satisfactory noise figure and by selectivity with respect to the amplitude of the input signals. A major disadvantage of field-magnetization cells is their considerable DC power consumption.

Ferrite-transistor cells are being further improved by increasing their speed and optimizing their design and operating characteristics. It is possible to increase the speed by using ferrite cores with a lower switching constant  $S_w$  and a smaller threshold  $H_0$ , by decreasing the size of the cores, and by using high-frequency semiconductor triodes. The operating characteristics of such cells can be optimized by improving their design, so as to /99 simplify building of the cells and units in which they are used and to increase their operating reliability.

The ferrite-transistor cell is usually made in the form of an integral module, which is considerably more convenient than using separate components of memory converter and triode which are then electrically connected into a circuit. The integral cells are checked during the manufacturing stage and are then used in assembling the various units. Basically, check-testing a finished cell, if both memory converter and triode are tested before assembly, consists in testing the parameters of the collector current pulse under load, by feeding write and read pulses to the windings  $w_w$  and  $w_r$  from square pulse generators.

## Section 20. Logic Circuits using Ferrite-Transistor Cells

In the construction of various logic circuits using ferrite-transistor cells, the selection can be confined to a few types of cells by means of which the basic AND, OR, NOT elements are realized (the operation NOT is performed by an inhibit cell) and to a master oscillator for pulses of clock frequency.

In all cases, when building units of ferrite-transistor cells, the basic elements of such cells, namely, the memory converter and the triode, must meet the following requirements: interchangeability of same-type elements, absence of any individual regulation of the elements as a function of load, identity of the rejection criteria for ferrite cores and triodes, stability of operation in an assigned range of temperature variation and feed voltage. Elements or combinations of elements whose action is based on time or amplitude relations between currents or voltages must be avoided. This makes extensive use of a field-magnetized cell undesirable in view of the fact that, because of the differences between the values of the coercive force of the cores, one must either

considerably increase the ampere-turns of the magnetizing field or individually adjust each cell. Aside from the labor involved in individual adjustment, this violates the principle of interchangeability of cells. Those versions of logic circuit design are preferable that do not require rigid synchronization of the pulses from the beginning of each cycle but which, within the limits of a single cycle, permit arbitrary delays of the code signals. This considerably facilitates the design of complex branched circuits.

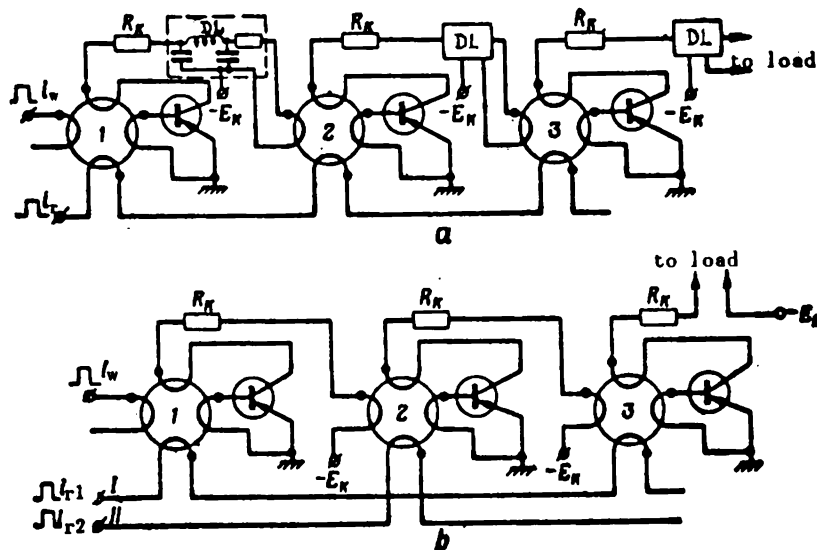


Fig.37 Series Connection of Ferrite-Transistor Cells  
a - Single-cycle circuit; b - Push-pull circuit

In units using ferrite-transistor cells it often becomes necessary simultaneously to write or read information into or from a large number of cells, /100 considerably exceeding the load-carrying capacity of an ordinary cell. In such cases, one uses either a cascade connection of ordinary cells increasing them to a number sufficient to switch all the cores of the load cells, or else high-power current amplifiers.

The most typical connection in logic circuits is the series connection of ferrite-transistor cells when the write winding of the core of the following cell is used as the load in the collector circuit of each preceding cell. Single-cycle, two-cycle and, to a lesser extent, three-cycle circuits for series connection of cells are in use.

As in similar circuits for ferrite-diode cells, in the single-cycle feed, the readout pulses (gating pulses) arrive simultaneously in the read windings of all cells of the circuit (Fig.37a). Under the action of the readout pulse, all cores in state 1 are flipped to state 0, and code transfer pulses arise at the output of the corresponding cells. These pulses, passing through a delay line composed of LC and RC circuits, set the cores of the following cells in the position 1. The purpose of the delay line is to delay the transfer pulse



arising in the collector winding of the core of the last cell until the end of the transients in the core of the next cell, caused by the action of the read-out pulse.

Push-pull circuits of series connection of ferrite-transistor cells are /101 more widely used (Fig.37b). The readout pulses are fed across two buses I and II, all odd-numbered cells being connected to one bus and all even-numbered cells to the other. The frequencies of  $i_{r1}$  and  $i_{r2}$  are the same, but the  $i_{r2}$  pulses are shifted a half-period with respect to the  $i_{r1}$  pulses. In contrast

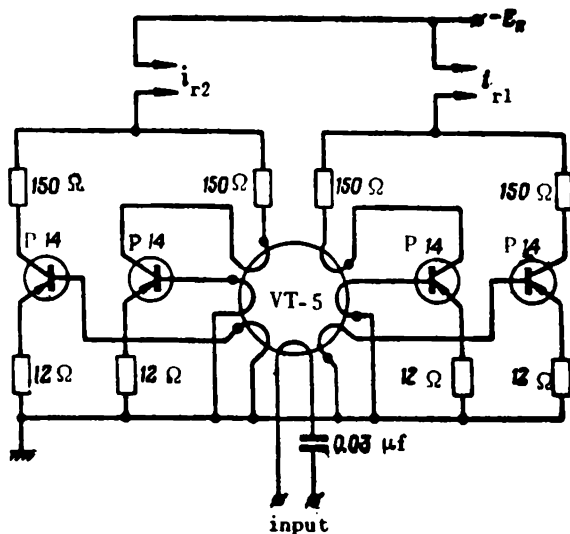


Fig.38 Generator of Two Series of Readout Pulses

to the single-cycle circuit, the arrival of a readout pulse, the readout of the code "1" from the preceding cell, and its writing into the core of the following cell all are simultaneous. In push-pull circuits, the operations of write and read of information from each cell are strictly separated in time. This improves their operating reliability at variations in the readout-pulse duration within fairly wide limits.

A single source - the generator shown in Fig.38 - may be used to form the two series of readout pulses  $i_{r1}$  and  $i_{r2}$ . The ferrite core of the generator transformer is of the VT-5 type,  $3 \times 2 \times 1.5$  mm, while the semiconductor triode is of a P14 type. The input winding of the core has 30 turns, and the other windings 12 turns each. The input winding is fed with sinusoidal oscillations whose DC component is generated by an  $0.03 \mu f$  capacitance connected across the input. During the positive half-period of the sinusoidal oscillations, the core is switched to the position  $+B_r$  and during the negative half-period, to the position  $-B_r$ . The right and left pairs of triodes are alternately opened, and at the output there arise readout pulses of the two series  $i_{r1}$ ,  $i_{r2}$  /102 which are shifted a half-period with respect to each other. Such a generator is a good matching element between a unit with a high output impedance (for

example a vacuum-tube oscillator) and a unit with a low input impedance.

The pulse generators used to give the operating clock frequency for units with ferrite-transistor cells may be any types of self-excited oscillators whose output stages are connected to the write and read windings of the cells. If stability of oscillator frequency is not particularly important and larger

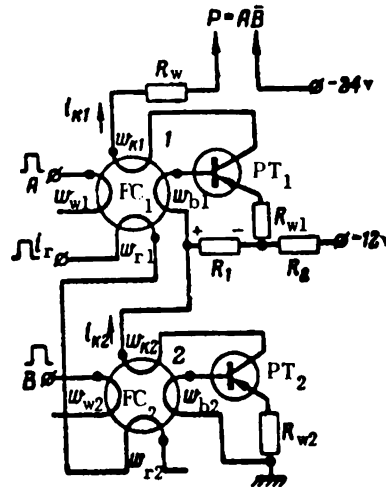


Fig.39 Negation Circuit with Negation at Various Times

pulse currents are required at the output, blocking oscillators in the self-oscillatory regime are used. The master oscillator used may also be a symmetric multivibrator using junction transistors to whose collector circuits the write and read windings of buffer ferrite-transistor cells are connected. If a more stable frequency of the gating pulses is required, generators of sinusoidal oscillations are used by means of which the multivibrators perform the function of matching circuits with the ferrite-transistor cells.

Let us consider some of the most widely used logic circuits based on ferrite-transistor cells.

NOT circuits. The logic relation realized by a NOT circuit has the form

$$P = A\bar{B},$$

where P is the output signal of the circuit;

A is the input code signal;

B is the input inhibit signal.

This relation is also realized by a NOT gate (Fig.34), one of whose disadvantages, as already noted, is the requirement for strictly simultaneous feeding of the pulses to the read and inhibit windings.

A NOT circuit with so-called different-time inhibition is used more widely; Fig.39 shows a practical version. The circuit is constructed with two cells that use VT-5 ferrite cores and PL5 triodes.

Core dimensions:  $FC_1 = 3.1 \times 2 \times 1.35$  mm;  $FC_2 = 3.1 \times 1.4 \times 1.35$  mm. 103  
Turn data:

$$w_{w1} = w_{w2} = 10;$$

$$w_{r1} = w_{r2} = 1;$$

$$w_{s1} = w_{k2} = 3;$$

$$w_{b1} = w_{b2} = 15.$$

Resistors:  $R_1 = R_3 = 33$  ohm;  $R_2 = 68$  ohm;  $R_{e1} = R_{e2} = 5$  ohm.

The code signal A is fed to the winding  $w_{w1}$  which writes 1 into the core  $FC_1$ , and the inhibit signal B is fed to the winding  $w_{w2}$ , likewise switching the core  $FC_2$  into position 1.

Each cell of a NOT circuit has its own source of collector feed. Here, if during a single cycle only the signal A or only the signal B is fed, the corresponding core is switched to the position 1 and then reset by the next regular readout pulse to the position 0. Then, the triode of the cell is opened and the current pulse  $i_{k1}$  or  $i_{k2}$  appears in the common collector circuit.

If, however, both signals - code signal A and inhibit signal B - are fed during a single cycle, then the cores  $FC_1$  and  $FC_2$  are at first switched to position 1 and are then reset by the readout pulse to position 0. In the readout period, however, only the triode  $PT_2$  is opened. The explanation is as follows: The current pulse  $i_{k2}$  causes a voltage drop across the resistor  $R_1$  connected in the base circuit of the first cell. This voltage drop compensates the emf of the base winding  $w_{b1}$  which is induced at the instant when the core  $FC_1$  is switched from position 1 to position 0. As a result, the triode  $PT_1$  remains blocked, the electrical circuit for the current  $i_{k1}$  is opened, and no signal appears at the output ( $P = 0$ ). For the same reason, we get  $P = 0$  if the following combination of signals occurs at the input:  $A = 0$ ,  $B = 1$ , i.e., only the inhibit signal is fed to the input. The code signal 1 appears at the output if and only if  $A = 1$ ,  $B = 0$ .

In a NOT circuit with different-time inhibition, the input signals A and B may be fed during the writing cycle, i.e., before the readout pulse, with any mutual phase shift. This is the fundamental advantage of such a NOT circuit.

Reliable inhibition of the action of the emf in the base winding  $w_{b1}$  by the voltage drop across the resistor  $R_1$  is possible if the following conditions are satisfied:

a) The duration of the current pulse  $i_{k2}$  must be longer than that of the current pulse  $i_{k1}$ . This is accomplished either by using cores of different size in the cells, as has been done in the circuit under consideration (the

cross section of the core  $FC_2$  is greater, and therefore its switching time  $\frac{104}{104}$  is also greater, meaning that the duration of the pulse  $i_{k2}$  is longer), or by connecting a second cell of greater resistance in the emitter circuit.

b) The operation of the second cell must begin about  $0.1 \mu\text{sec}$  before that of the first cell. To meet this condition, triodes must be rejected according to the trigger delay time.

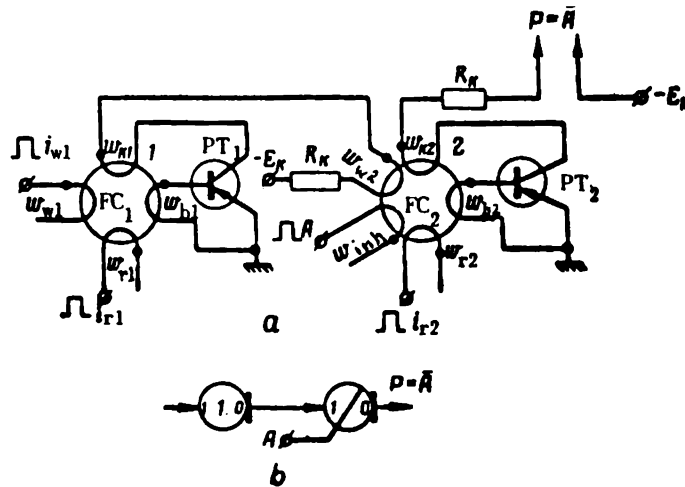


Fig.40 Inverter

a - Principal wiring diagram; b - Symbolic representation

In accordance with the character of the processes taking place in the NOT circuit during the readout cycle, such a circuit is predominantly a voltage circuit. Here it is only necessary to ensure that one quantity exceeds the other (the voltage drop across the resistor  $R_1$  must be greater than the emf in the core  $w_{b1}$  when the core  $FC_1$  is switched from position 1 to position 0; in addition, various degrees of excess are allowable. In this lies the main difference from all other types of compensating circuits, in which the parameters must be equal for various operating regimes, which is naturally considerably more difficult.

The principal drawback of this NOT circuit is the need for two sources of collector feed. It must also be borne in mind that normal operation of the circuit requires sufficiently powerful readout current pulses to switch the core without a supplementary magnetic field due to positive feedback. This becomes necessary at  $A = 1$  and  $B = 1$ .

The inverter. The inverter is used for the operation of logic negation. The inverter circuit consists of two ferrite-transistor cells (Fig.40).

The first cell of the inverter is the ones generator. This is essentially a ferrite-transistor cell to whose write winding  $w_{w1}$  and read winding  $w_{r1}$  the write and read pulses are continuously and alternately fed. Consequently,

at the output of the ones generator, a sequence of pulses coinciding in time with the readout pulses is formed. The ones generator is symbolically represented in the form of a circle with a one inside it (Fig. 40b). The second cell of the inverter is the inhibit cell. Its write winding is fed with pulses from the ones generator, while the code signal A goes into its inhibit winding.

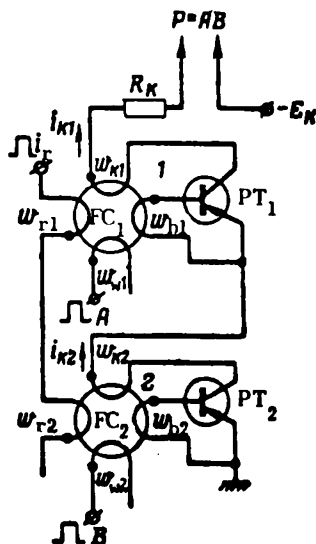


Fig. 41 Coincidence Circuit using Ferrite-Transistor Cells

For  $A = 0$ , pulses from the ones generator are transformed by the inhibit cell at the output of the inverter, i.e.,  $P = 1$ . On the other hand, if  $A = 1$ , i.e., if a sequence of pulses coinciding in time with the pulses from the ones generator is fed to the inhibit winding, then there will be no pulses at the output ( $P = 0$ ). Thus the inverter is a dynamic element at whose output a sequence of pulses either does appear ( $P = 1$ ) or does not appear ( $P = 0$ ).

Coincidence circuit. Several versions of building coincidence circuits from ferrite-transistor cells are possible. In one of these versions, the coincidence circuit is built of three cells, like a circuit with ferrite-diode cells (Fig. 30), i.e., using inhibit cells. The practical application of this version is limited by the requirement for strict time synchronization of the input signals as well as by the considerable power consumption of the cells.

The version of the coincidence circuit shown in Fig. 41 is more widely used. This circuit consists of simple ferrite-transistor cells connected in parallel. The number of these cells equals the number of input quantities.

The circuit operates in push-pull: during the write cycle, the input code signals A and B are fed and during the read cycle, the readout current pulse  $i_r$ . If only a single input signal is given, for example the signal A, then the core FC<sub>1</sub> is switched to position 1. During the readout cycle, the core is reset to position 0, but the triode PT<sub>1</sub> remains cut off (it is opened only on the

base) since the circuit for its collector current is open (the triode  $PT_2$  is cut off). Consequently, the switching of the core  $FC_1$  from position 1 to position 0 is accomplished only by the readout current, without participation of the additional field produced by the positive feedback of the cell. The 106 circuit operates similarly if only the signal B is fed to the input, i.e., when  $A = 0$ ,  $B = 1$  we also get  $P = 0$ .

If both input signals are fed during the write cycle ( $A = 1$ ,  $B = 1$ ), then the cores  $FC_1$  and  $FC_2$  are set in position one. During the readout cycle, both cores are switched to the position zero, the triodes  $PT_1$  and  $PT_2$  are opened, the circuit for the collector current is closed, and the code signal 1 appears at the output of the circuit.

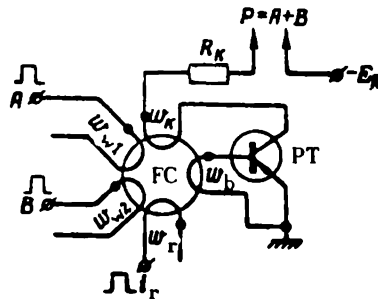


Fig.42 Collector Circuit

Stable operation of the circuit under various temperature conditions is ensured if the following requirements are met:

- a) The number of readout ampere-turns must be sufficient to switch the cores without participation of the positive feedback of the cells.
- b) The switching time of the cores of the cells must be the same.
- c) The readout pulse repetition rate must be so selected that its period is longer than the resorption time of the non-base carriers of the base current of one triode, with the second triode cut off. The meaning of this requirement is as follows: Let only the signal A be fed during the write cycle. Then, during the readout cycle, the magnetic state of the core  $FC_2$  will not change (the code "0" will be held in it), the triode  $PT_2$  will be blocked, the core  $FC_1$  will be switched into the zero position, and the triode  $PT_1$  will be opened only at the base on account of the emf in the winding  $w_{b1}$ . The base of the triode  $PT_1$  will be saturated with non-base carriers, which take a rather long time for resorption because of the slightly controlled collector current of the triode  $PT_2$ . Before resorption of the non-base carriers has been completed, the triode  $PT_1$  is open on the base. During the following write cycle, let only the signal B, which writes 1 into the core  $FC_2$ , be fed in and let the resorption of the non-base carriers in the base of the triode  $PT_1$  not be completed on arrival of the readout pulse. This will lead to the appearance of a spurious signal at the output of the circuit. The magnitude of this signal may be comparable with

that of the code signal 1. Thus, the need to decrease the readout pulse repetition rate leads to a decrease in the speed of such circuits.

With increasing number of inputs in a coincidence circuit, its load-carrying capacity decreases, i.e., the output signal of the circuit will be able to write or read information only from a smaller number of ferrite-transistor cells. This is due to the decrease in the output current produced by the overall voltage drop across the series-connected collector windings of the coincidence circuit. The necessary load-carrying capacity of a coincidence circuit with a large number of inputs is maintained by decreasing the number of turns in the collector windings of the cells. /107

Collector circuit. The operation of logical addition is realized by a collector circuit which in its most elementary form consists of a simple ferrite-transistor cell with the same number of write windings as there are input quantities. The circuit presented in Fig.42 realizes the logic relation  $P = A + B$ . If only a single signal is fed to the input, the core is switched to position 1, and is then reset by the readout pulse to position 0, causing the code signal 1 to appear at the output.

Nonequivalent circuit. A circuit of logic nonequivalence (OR - OR circuit) realizes the logic function

$$P = \overline{A}\overline{B} + \overline{A}B. \quad (36)$$

The code signal 1 appears at the output of a nonequivalent circuit if and only if one of the input signals is fed to the inputs, i.e., if and only if the following sets of input variables are present:  $A = 1, B = 0$  or  $A = 0, B = 1$ .

In designing an OR - OR circuit, it is convenient to represent the function (36) in the form

$$P = \overline{A}\overline{B}(A + B). \quad (37)$$

The functions (36) and (37) are equivalent, which is easy to prove. In fact, we have

$$\begin{aligned} P &= \overline{A}\overline{B}(A + B) = (\overline{A} + \overline{B})(A + B) = \overline{A}A + \overline{A}B + A\overline{B} + B\overline{B} = \\ &= \overline{A}B + A\overline{B}. \end{aligned}$$

The functional and principal wiring diagrams of nonequivalence, realizing the relation (37), are given in Fig.43. The circuit is constructed of three ferrite-transistor cells, the first cell performing the operation OR, and the second and third the operation AND. All three cells participate in performing the operation NOT.

At first, the cores of all cells are in the zero position. If, during the write cycle, both input signals are fed ( $A = 1, B = 1$ ), all cores of the cir-

cuit will be switched to the position one. During the readout cycle, the cores are returned to the zero position, and only the triodes  $PT_2$  and  $PT_3$  are completely open. The triode  $PT_1$  remains in the cut-off state and is open only on the base. This is due to the fact that the collector current  $i_{k2}$  arising on simultaneous switching of the cores  $FC_2$  and  $FC_3$  to the zero position, produces across the resistor  $R_1$  a voltage drop compensating (suppressing) the emf in the base winding of the core  $FC_1$  during its switching to position 0, thereby keeping the triode  $PT_1$  in the cut-off state. Consequently, there will be no /108  
signal at the output ( $P = 0$ ).

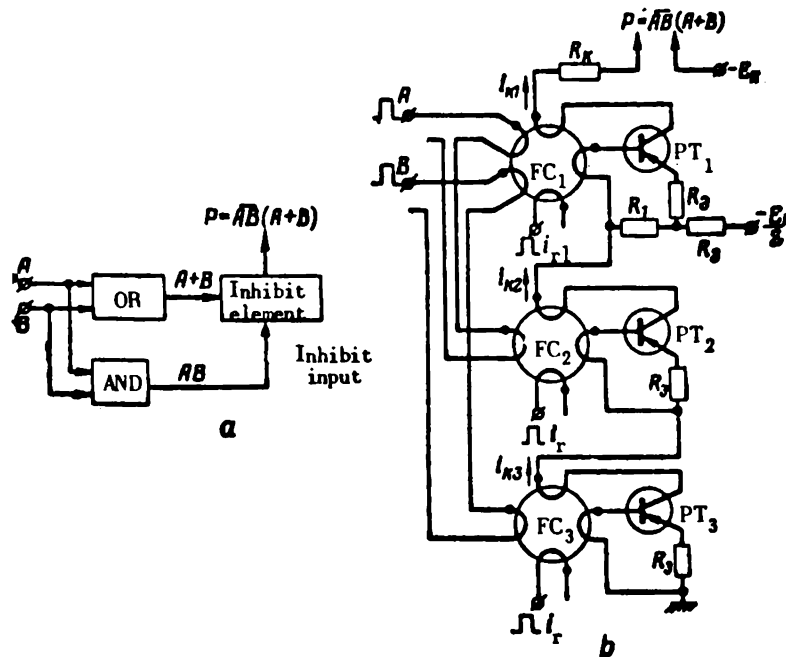


Fig. 4.3 Nonequivalent Circuit  
a - Functional diagram; b - Schematic diagram for  
ferrite-transistor cells

If, during the write cycle, only one input signal is given, then either the cores  $FC_1$  and  $FC_2$  (at  $A = 1, B = 0$ ) or all cores  $FC_1$  and  $FC_3$  (at  $A = 0, B = 1$ ) are switched to position 1. During the readout cycle, the initial position of these cores is restored, and only triode  $PT_1$  is completely open. Triodes  $PT_2$  and  $PT_3$  remain in the cut-off position, since the electric circuit for their collector currents remains open. In this case, the code signal 1 appears at the output.

Here, as in coincidence circuits (Fig. 4.1) and inhibit circuits (Fig. 3.9), there must be enough readout ampere-turns to switch the cores without participation of the positive feedback in the cells.

A nonequivalent circuit may be used to form the complement code of a number from the direct code. For this purpose, the code pulses of the direct code



of the number are successively fed to one input of the circuit, while the gating pulses, synchronously with the code pulses and at the same frequency, are fed to the second input. The code pulses of the inverse code are formed at the output of the circuit.

Section 21. Basic Logic Elements from Electron Tubes and Semiconductor Devices. Circuits with Direct Connections /109

Vacuum tube circuit of the logic elements. In constructing logic elements from electron tubes, either voltage or current pulses or potential levels can be used as the code signals at the inputs and outputs.

The logic coincidence circuit for two or more inputs can be constructed

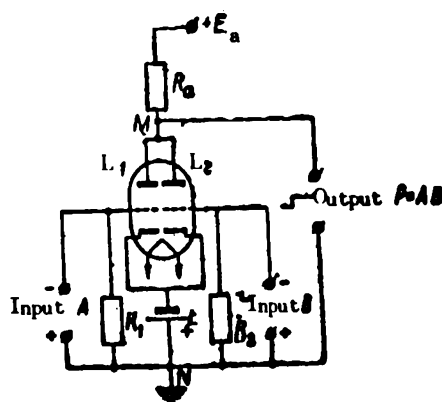


Fig. 44 Potential Coincidence Circuit with Two Inputs, using a Duo-Triode Tube

from single-grid or multigrid electron tubes. Figure 44 gives a potential coincidence circuit with two inputs using a duo-triode tube. In the absence of input signals A and B, both tubes  $L_1$  and  $L_2$  are open. The internal resistance of the open tubes is many times smaller than the plate resistance  $R_p$ . For this reason, the voltage  $E_a$  is mostly applied to the resistor  $R_4$  where the output will be at a low potential level corresponding to the code "0".

The code signal 1 at the input is represented by a low potential level and the code signal 0 by a high level. If only a single input signal A or B is fed, the corresponding tube is cut off. Even in this case, however, the internal resistance of the open tube is, as before, less than the plate resistance  $R_p$ , i.e., as before there will be a low potential level (code "0") at the output. If the two input signals 1 are fed simultaneously, then the tubes  $L_1$  and  $L_2$  will be cut off, and the voltage at the output will rise to the value  $E_a$ , which corresponds to the code "1".

The positive bias voltage  $+E_g$  on the grid of both tubes is imposed in order still more to decrease the resistance of part of the circuit M - N in the

absence of input code signals.

The pulse coincidence circuit with two inputs (gates) using the pentode 6Zh2P with two control grids, has found wide practical application (Fig.45). Gates of this type are particularly advisable in cases where the input code signals are difficult to synchronize. Since the code and control signals at the inputs of the circuit do not coincide in time in the general case, a small memory capacitance is connected to one of them, which remembers the code signal that has arrived at one input until the time of arrival of a code signal at the other input. The memory time is about  $1 \mu\text{sec}$ . The normal operation of the circuit requires the signal A to arrive not later than the signal B. On termination of the code pulses A and B, a clearing pulse is fed to the capacitor  $C_1$ , to the zero position  $U^{00}$ , coinciding in time with one of the synchronizing (gating) pulses.

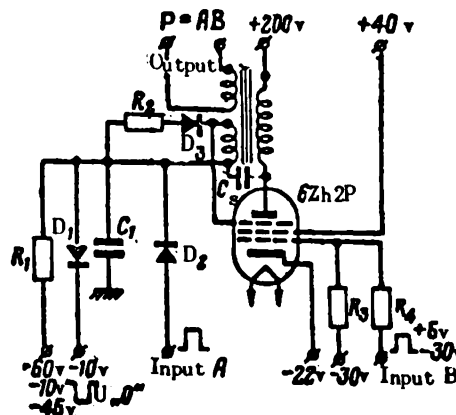


Fig.45 Vacuum-Tube Gate with Memory Capacitance

In the absence of code signals, the tube is cut off on the first and third grids ( $E_{g1} = -8 \text{ v}$ ,  $E_{g3} = -16 \text{ v}$ ). The code signal A consists of a positive pulse of 30 - 35 v amplitude fed from the level of -38 v across the diode  $D_2$ . The signal A charges the memory capacitance  $C_1$  to a potential of the order of -10 v and thus opens the pentode on the third grid by +12 v relative to the cathode. The rise of the potential on the capacitor by more than -10 v is limited by the diode  $D_1$  connected to the -10 v source. After feeding only the signal A, no plate current will flow through the tube since this latter is still closed on the first grid. On arrival of the signal B, likewise represented by a positive pulse of 30 - 35 v amplitude (from the level -30 v), the tube is open on both grids and the code signal 1 (signal P) appears at the output.

The positive feedback in the circuit of the third grid of the pentode forces its triggering by the third grid.

The current of the third grid discharges the capacitor  $C_1$ . However, the presence of stray capacitance  $C_s$  between the windings of the output transformer causes charging of the capacitor  $C_1$ . Consequently, at the end of the input

code signals, the circuit may also fail to return to the initial state and, on arrival of only a single regular signal B, again produce a pulse of the code "1" at the output, but this time a spurious signal. This undesirable phenomenon is eliminated by feeding to the memory capacitor  $C_1$  a clearing series of negative erase pulses  $U^{*0}$  of 30 - 35 v amplitude (from the level -10 v). The pulses  $U^{*0}$  every 1.5  $\mu$ sec will discharge the capacitor  $C_1$  to the potential -38 v.

The overshoot of the pulses in the secondary of the transformer is damped by the aid of the diode  $D_3$  and the resistor  $R_3$ . In addition to the leak resistor  $R_3$ , the circuit of the first grid also includes the resistor  $R_4$ , serving to stabilize the power dissipation of the grid. The resistor  $R_1$ , connected 111 to the -60 v source, is designed to maintain a potential of -38 v on the memory

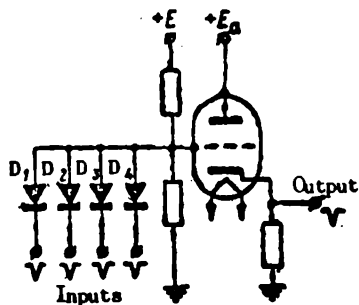


Fig.46 Pulse Common Collector Circuit for Signals of Negative Polarity

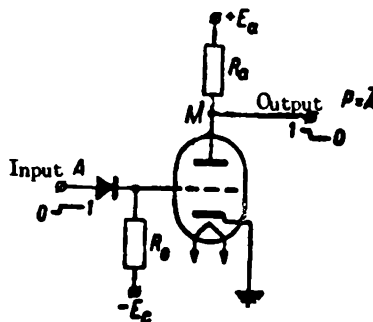


Fig.47 Inverter using a Vacuum-Tube Triode

capacitor, in the absence of input code signals. This coincidence circuit using a pentode is convenient because of the fact that, thanks to the pulse transformer, it permits pulses of the desired polarity to be obtained at the output. The output of the circuit can also be loaded across a small resistance,

Collector circuits can often be obtained from coincidence circuits by simply changing the operating states of such circuits. For example, if instead of a positive bias a negative bias is applied to the grid of the tubes in a potential coincidence circuit (Fig.44) in such a manner that in the absence of input code signals both tubes are held in the blocked state, and if high-level potentials are used as the code signal 1 at the input, then if a code signal 1 is present at at least one of the inputs, the corresponding tube ( $L_1$  or  $L_2$ ) will be opened and the low-level signal (code "1") will appear at the output.

Figure 46 gives a pulse common collector circuit for signals of negative polarity, using a triode tube and semiconductor diodes. The diagram shows four inputs, but their number may be either increased or decreased. On appearance of a code pulse of negative polarity at any of the inputs (or on several inputs simultaneously), the tube is cut off and a negative pulse (the code signal 1) appears at the output of the circuit. The diodes perform purely separating functions, eliminating mutual inductance of the input circuits.

An inverter may be built of a single triode (Fig.47). If there is a low level potential corresponding to the code "0" at the input, then the tube is blocked by a negative bias  $-E_g$  applied to the grid across the bias resistance. In this case, the plate potential equals the source potential  $+E_a$ , i.e., there will be a high level signal at the output, corresponding to the code "1". On application of a high potential (code "1") to the input A, the tube is opened /11/ and the plate potential falls, causing the code signal 0 to appear at the output. Consequently, the inverter realizes the logic negation  $P = \bar{A}$ .

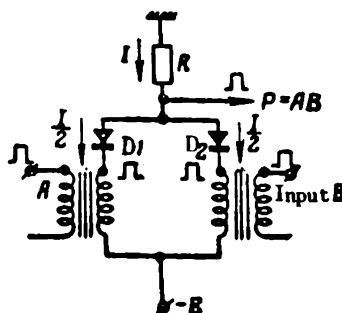


Fig.48 Diode-Rheostat Coincidence Circuit

Diode-rheostat and diode-transformer circuits. Semiconductor diodes are widely used to construct logic circuits, since they have a longer service life than vacuum tubes, and are smaller and more economical.

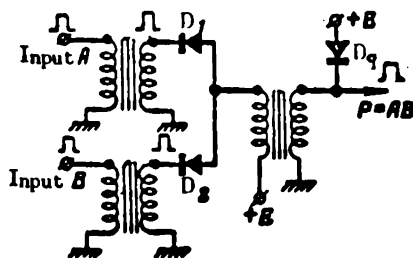


Fig.49 Diode-Transformer Coincidence Circuit

Two types of logic circuits based on semiconductor diodes are used: diode-rheostat and diode-transformer. Diode-rheostat circuits are composed of diodes and resistors. They are simple and compact, but require amplification of the signals by the work of one circuit on another.

The principal elements of diode-transformer circuits are diodes and pulse transformers, for which purpose toroidal transformers with ferrite cores can be used. Such circuits require no intermediate amplifiers for each stage. A change in the diode characteristics has no effect on the parameters of a diode-

transformer circuit. This is due to the fact that the diodes are connected between the windings of the transformers. The forward resistance of the diode is much less than the impedance of the winding for the AC component of the pulse (the code signals being represented by pulses). The back resistance of the diode, however, is considerably greater than the resistance of the winding for the DC component of the signal. The main drawback of diode-transformer circuits is their sensitivity to the phase relations of the signals, which decreases the operating reliability.

In a diode-rheostat coincidence circuit with two inputs (Fig.48), the transformers are the output transformers of the write elements putting out the code pulses A and B. In the absence of input code signals ( $A = 0, B = 0$ ), both diodes are open allowing passage of equal currents  $\frac{1}{2}I$ , while there will be a low potential (code "0") at the output. If only a single code signal is fed ( $A = 1, B = 0$  or  $A = 0, B = 1$ ), one of the diodes will be open and all the

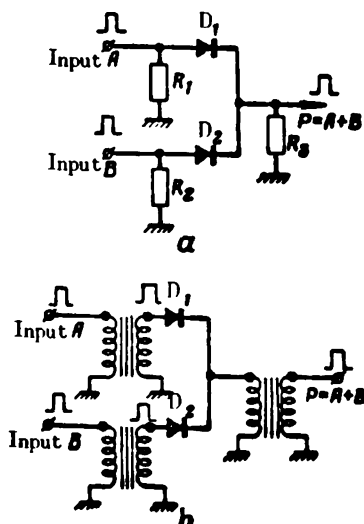


Fig.50 Common Collector Circuits  
a - Diode-rheostat type; b - Diode-transformer type

current will be redistributed in the branch of the other diode. In this case, the current through the resistor R is practically unchanged, since  $R \gg R_1$  (where  $R_1$  is the internal resistance of the open diode). Consequently, we get  $P = 0$  also in this case. Finally, when positive code pulses arrive simultaneously at both inputs, the diodes  $D_1$  and  $D_2$  are closed, the current is interrupted, and a positive voltage pulse (code "1") appears at the output. 113

Figure 49 gives a coincidence circuit of the diode-transformer type. The logic operation AND is performed by the aid of the diodes  $D_1$  and  $D_2$ , while the transformers perform the functions of input and output circuits. The input code signals of positive polarity block the diodes  $D_1$  and  $D_2$ . A positive pulse at the output, corresponding to the code "1", appears if and only if the sig-

nals A and B are simultaneous. The quenching diode  $D_q$ , connected in parallel with the output circuit, increases the ratio of the useful signal 1 to the noise. The resistance of the quenching diode is low for weak signals and high for strong signals. In circuits operating on the principle of signal coincidence, the noise in power is weaker than the useful signal 1 at the output, since at the instant of coincidence the powers are summated.

Figure 50 gives examples of the arrangement of common collector circuits of diode-rheostat and diode-transformer types. In these circuits, the positive code signal "1" appears at the output when the code 1, represented by a pulse of positive polarity, arrives at at least one of the inputs. The diodes perform the function of separating the input circuits.

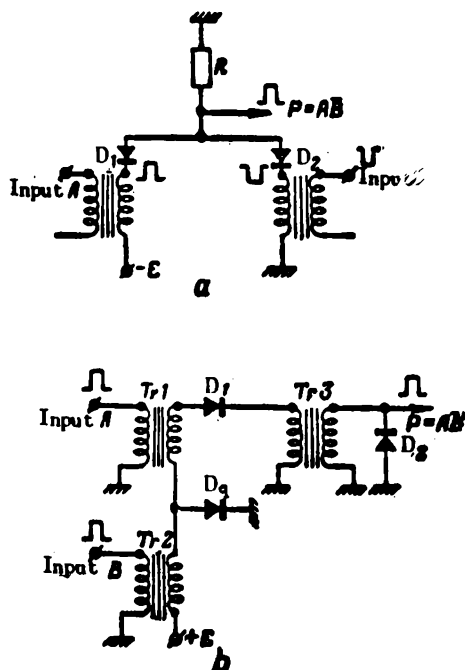


Fig.51 Inhibit Circuits

a - Diode-rheostat type; b - Diode-transformer type

A diode-rheostat inhibit circuit (Fig.51a) is readily obtained from a coincidence circuit (Fig.48) by making certain changes in its operating regime. The secondary winding of the right pulse transformer is grounded, and the inhibit signal B is represented by a pulse of negative polarity. In the absence of input signals ( $A = 0$ ,  $B = 0$ ), the diode  $D_1$  is open while the diode  $D_2$  is cut off. Current flows in the left branch of this circuit across the diode  $D_1$ . If, simultaneously with the signal A, the inhibit signal B is applied ( $A = 1$ ,  $B = 1$ ), the diode  $D_1$  is cut off and the diode  $D_2$  is open; current will flow in the right branch of the circuit across the diode  $D_2$ , and at the output of the circuit - as before - the code signal "1" is absent. In the combination  $A = 1$ ,  $B = 0$ , i.e., when the code signal 1 is fed only to the input A, the diode  $D_1$  is

cut off and the positive pulse corresponding to the code "1" appears at the output. Thus, the circuit realizes the relation  $F = AB$ .

In the diode-transformer inhibit circuit (Fig.51b), the secondaries of the transformers Tr1 and Tr2 are connected in tandem, and the quenching diode  $D_q$  is connected to the center tap of these windings. On account of the series connection of the secondary windings, the code signal A is compensated by the /114 inhibit signal B. In the absence of an inhibit signal ( $B = 0$ ), the lower end of the secondary winding of the transformer Tr1 is below the potential of the ground. For reliable compensation of the signal A it is necessary that the amplitude of the pulse in the output winding of the transformer Tr2 be slightly higher than that of the pulse in the output winding of the transformer Tr1.

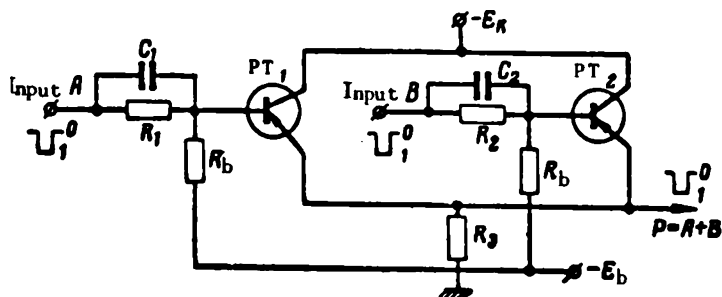


Fig.52 Common Collector Circuit for Signals of Negative Polarity

For stable operation of a diode-transformer inhibit circuit, the inhibit signal B must be applied before the code signal A. The strict conditions for synchronization of the input signals of the circuit are its main drawback.

Logic elements with semiconductor triodes. Such triodes have promoted the manufacture of a compact, economical, and reliable computer. The use of triodes with various types of conductivity (p-n-p and n-p-n) gives the designer a greater opportunity to improve circuits of high-speed digital computers than does the use of electron tubes. In designing logic circuits based on semiconductor triodes, predominantly emitter followers and inverters are used.

The logic elements for realizing the basic logic relations AND, OR can be formed by connecting emitter followers in parallel with type p-n-p triodes. /115 The collector circuit with two inputs shown in Fig.52 consists of two emitter followers. In the absence of signals at the inputs, the triodes PT<sub>1</sub> and PT<sub>2</sub> are in a weakly conducting state, and the voltage at the output is approximately equal to the zero ground potential which corresponds to the code "0". If the code signal 1 of negative polarity is fed to at least one input of the circuit, the corresponding triode is opened and the increasing current of the triode lowers the potential at the output, in consequence of the voltage drop across the resistor R<sub>3</sub>. Thus, the code signal 1 of negative polarity appears at the output. By changing the operating conditions and parameters of several elements, this circuit can be used for the operation of logical multiplication.

Despite the frequent use of emitter followers for the construction of logic circuits, such circuits are not free from faults. The main cause of these faults is the absence of voltage amplification in the emitter follower.

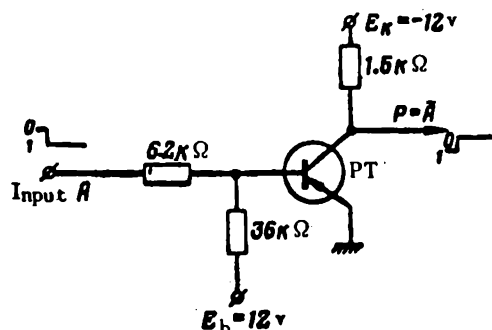


Fig.53 Inverter using a Semiconductor Triode

In the circuit of an inverter using a semiconducting triode with rheostat connections (Fig.53), the feed is provided by the voltages  $E_k$  and  $E_b$ . The magnitudes of the resistances are so selected that the output signal of the inverter can be used to control several similar inverters. The input voltage

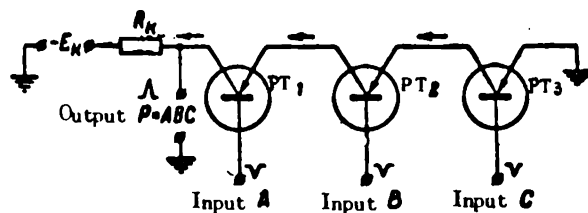


Fig.54 Coincidence Circuit using Direct-Coupled Triodes for Negative Input Signals

divider ensures the nonconducting state of the triode at zero potential of the input signal, corresponding to the code "0", and also maintains the required 116 value of the base current on application of a signal of negative polarity corresponding to the code "1". At  $A = 0$ , the triode is cut off and there will be a low potential approximately equal to  $-12\text{ v}$  ( $P = 1$ ) at the output. At  $A = 1$ , the triode is opened and the potential at the input rises to almost zero, i.e.,  $P = 0$ .

Direct-coupled logic circuits. Semiconductor triodes permit the construction of direct-coupled circuits. A coupling between the collector of one triode and the base (collector or emitter) of the other is called direct if it has no intermediate elements.

Direct-coupled circuits were first used after the development of surface-



barrier triodes which, in their parameters and characteristics, proved to be most suitable for the construction of such circuits. The triodes here are connected by a circuit with grounded emitter, either in series or in parallel.

A surface-barrier transistor is a semiconducting triode in which two rectifying electrodes or contacts are laminated, by a special electroplating process, to opposite sides of a thin electrolytically polished sheet of germanium. Such triodes are characterized by a base thickness of about  $5\ \mu$  and a cutoff frequency from 30 to 80 Mc.

The most widely used Soviet surface-barrier triodes are types P404, P405.

Let us consider examples of the construction of direct-coupled logic circuits.

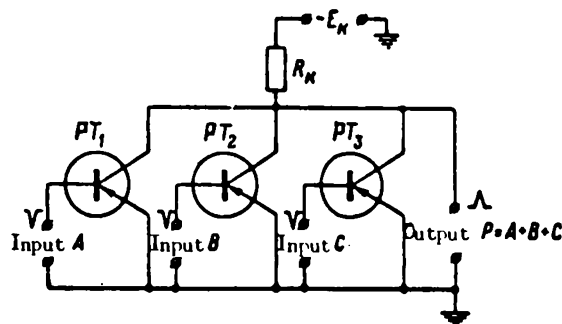


Fig.55 Direct-Coupled Collector Circuit with Triodes for Negative Input Signals

A logic coincidence circuit with three inputs for negative input signals (Fig.54) realizes the relation  $P = ABC$ . The triodes are connected in series. In the absence of signals at the inputs, the triodes are blocked and the voltage at the output is close to the voltage of the feed source  $-E_K$ . When pulses of negative polarity, representing the code "1", are fed simultaneously to all the inputs, the triodes are open during the time of action of the pulses, and a relatively large collector current flows through the series circuit of the triodes and the resistor  $R_K$ . The potential at the output of the circuit decreases from  $-E_K$  to almost zero, i.e., a positive voltage pulse appears at the output. Consequently, this circuit is a coincidence circuit with phase reversal for negative input signals. It can be used as a collector circuit with phase reversal for positive input signals. In that case, in the absence of input signals, all the triodes are open and a potential close to zero will appear at the output. On arrival of a positive signal at at least one input, the corresponding triode is cut off, the circuit for the collector current is broken, and a signal of negative polarity appears at the output.

With parallel connection of the triodes, a directly coupled logic circuit can operate either as a coincidence circuit or as a common collector circuit with simultaneous phase reversal of the input signals. The circuit shown in Fig.55 is a common collector circuit with phase reversal for negative input

signals. This circuit realizes the relation  $P = A + B + C$ .

In the absence of input signals, the triodes are cut off, a very small collector current flows through the resistor  $R_k$ , and the output voltage is practically equal to the feed voltage  $-E_k$ . On appearance of a negative pulse at any input, the corresponding triode is opened and a positive voltage pulse appears at the output. The same circuit can perform the functions of a coincidence circuit with phase reversal for positive input signals.

Direct-coupled logic circuits have the following advantages:

- a) No intermediate coupling circuits are present, and only triodes and resistors are used. Owing to the absence of transformers and capacitors, such circuits are distinguished by simple design, and printed circuitry is readily feasible.
- b) The same logic element can be used either as a coincidence circuit or as a common collector circuit.
- c) Only a single source is required to feed the circuit, with a feed voltage of about 3 v. The power dissipation in triodes and resistors is /118 very small. Very little power is used in charging and discharging stray capacitances, since the voltage drops in direct-coupled circuits are of the order of 0.4 v.
- d) On account of the use of high-frequency triodes and small voltage drops, high operating speed is attained.

The disadvantages of direct-coupled logic circuits are as follows:

- a) low economy, since the number of triodes in the coincidence circuits and collector circuits is proportional to the number of inputs;
- b) insufficient utilization of the frequency capabilities of triodes, since they operate in a regime of deep saturation;
- c) need for careful selection of the triodes with respect to many parameters. The allowable range of the triode parameters in a circuit is 3 - 5%.

## Section 22. Construction of Compound Logic Circuits

Compound logic circuits are constructed from elementary logic circuits, which are termed logic elements. In the build-up of logic circuits it is first necessary to select the types of logic elements and then to determine the simplest structure of the circuit, i.e., a method of connecting the elements such that their number will be minimum. These problems are solved by the aid of mathematical logic.

As already noted, the fundamental functionally complete system of logic functions is preferable for the synthesis of logic circuits for digital com-

puters. The technical realization of this system is accomplished by the aid of the basic logic elements AND, OR, and NOT. From the elements AND, OR, and NOT, any logic circuit as complex as may be desired can be constructed. Logic circuits constructed from logic elements connected in a certain definite way, are also called function circuits.

To construct the function circuit of some unit, its operating conditions must be defined and formulated, i.e., the logic operators that the circuit is to realize must be determined and written in the form of a logic function. It is convenient to write out the operating conditions of the circuit in the form of a truth table, which indicates the signal (1 or 0) that will appear at the output of the circuit for certain combinations of code signals at the inputs. From such a truth table, it is easy to set up a logic function (which is a formula written in the symbolic form used in logical algebra) of the operating conditions of the logic circuit, the composition of its basic elements, and the order of their interconnection. The logic function, before using it for laying out a function circuit, is subjected to maximum simplification.

The sequence of connection of the logic elements of a function circuit is determined by the sequence of execution of the operations in the expression /119 for the logic function. Here we must bear in mind that the performance of the operations AND, OR, NOT in an expression for a logic function is accomplished in the same order in which the ordinary arithmetic operations are performed, taking the operation AND to be equivalent to multiplication and the operation OR to be equivalent to addition. A general negation of some expression is equivalent to enclosing this expression in parentheses.

### Establishing a Logic Function from a Truth Table

Two methods of setting up a logic function from the values of an assigned truth table are in use:

- 1) from the conditions of truth, or from the ones;
- 2) from the conditions of falsehood, or from the zeros.

We can illustrate this on hand of examples. Let it be required to derive logic functions with three units each, having two inputs (A and B) and one output ( $P_1$ ,  $P_2$ , and  $P_3$  respectively). These units operate according to a truth table (Table 12).

To set up a logic function from the conditions of truth (from the ones), each line of the truth table must be represented in symbolic form for which, for the given unit, the code "1" appears at the output. For example, for the first unit there is only one line where  $P_1 = 1$ . The conditions of operation of this unit are formulated as follows: The code signal 1 appears at the output of  $P_1$  when there are no signals at the two inputs. Consequently, in symbolic form

$$P_1 = \bar{A}\bar{B}. \quad (38)$$

TABLE 12

## OPERATING CONDITIONS OF UNITS

<i>A</i>	<i>B</i>	<i>P</i> <sub>1</sub>	<i>P</i> <sub>2</sub>	<i>P</i> <sub>3</sub>
1	1	0	0	1
1	0	0	1	1
0	1	0	1	1
0	0	1	0	0

For the second unit there are two lines where  $P_2 = 1$ , and the conditions for its operation are formulated as follows: a signal appears at the output 120 of the unit if there is a signal at the input A and no signal at the input B or, on the other hand, if there is a signal at the input B and no signal at the input A, i.e.,

$$P_2 = A\bar{B} + \bar{A}B. \quad (39)$$

Finally, the third unit operates as follows from the conditions of truth: A signal appears at the output if there is a signal at both inputs, if there is the signal A but not the signal B, or if there is the signal B but not the signal A. Consequently,

$$P_3 = AB + A\bar{B} + \bar{A}B. \quad (40)$$

Based on eqs.(38) - (40), we can formulate the following rule for setting up a logic function from the ones: for each line of the truth table in which the output signal is one, we set up the product of the literal representations of the input signals (if the value of the input signal is zero, we take the negation of the literal representation of that signal) and then add these products.

To set up a logic function from the conditions of falsehood (from the zeros) we represent in symbolic form each line of Table 12 in which the output signal is zero for the given unit.

The operation of the third unit, from the conditions of falsehood (last line of Table 12), is written as follows:

$$\bar{P}_3 = \bar{A}\bar{B}, \quad (41)$$

i.e., there is no signal at the output if there are no signals at both inputs.

To pass to the true proposition, i.e., to obtain the conditions for which

a signal corresponding to the code "1" appears at the output of the third unit, we must perform the operation of negation of the left and right sides of eq.(41). Then, by the law of inversion, we get

$$P_3 = \overline{\bar{A}\bar{B}} = A + B. \quad (42)$$

The second unit, according to the conditions of falsehood, operates as follows (first and fourth lines of Table 12): there is no signal at the output when there are either signals at both inputs or no signals at both inputs. Consequently,

$$\bar{P}_2 = AB + \bar{A}\bar{B}.$$

Performing the negation of the left and right sides, we get

$$P_2 = \overline{AB + \bar{A}\bar{B}} = (\bar{A} + \bar{B})(A + B). \quad (43)$$

For the first unit, the logic function obtained in the same way, is of the form

$$\bar{P}_1 = AB + A\bar{B} + \bar{A}B$$

or

$$P_1 = (\bar{A} + \bar{B})(\bar{A} + B)(A + \bar{B}). \quad (44)$$

On the basis of eqs.(42) - (44), we formulate the following rule for /121 setting up a logic function from the zeros: For each line of the Table of operating conditions in which the output signal is zero, we set up the sum of the literal representations of the input signals, taken with negation, and then multiply these sums.

In setting up a logic function we must use a rule that permits obtaining the simplest notation for the formula. For example, in analyzing Table 12, it can be said that for the first unit it is better to set up a logic function by ones, since there is only one line for which  $P_1 = 1$ . In fact, a comparison of eqs.(38) and (44) shows that these are logically equivalent since they express, in the last analysis, one and the same logical problem. Each of these formulas, however, represents different physical systems, with the system represented by eq.(44) being far more complex. It should be borne in mind that, after simplifying eq.(44) by the aid of the basic laws of logical algebra, it reduces down to eq.(38).

The simpler the logic function, the less complex will be the function circuit of the logic unit that realizes this function. A circuit is customarily considered as being simplest if it has the smallest number of elements and if the elements themselves are very simple or are composed only of elements with two inputs or only of elements of the same kind.

Thus, to obtain the least complex function circuit, the logic function

must be simplified.

Each transformation of a logic function means a multiplication in the physical system realizing it, without changing the interrelations between the input and output signals. For this reason, logic functions are highly convenient, not only to represent the operation of logic circuits without drawing them in detail but also to modify the circuit design during the search for the most acceptable version.

Order of the construction of function circuits. Let us consider how a function circuit is constructed from a prescribed logic function. The starting factors may be the following: In an assigned logic function, reduced by transformation to the simplest form, the entire sequence of operations to be performed is broken down into stages. Each stage comprises those logic operations that can be performed independently of each other. Consequently, the operations contained in a single stage can be performed in any order. The number of such stages will be termed the order of the logic function.

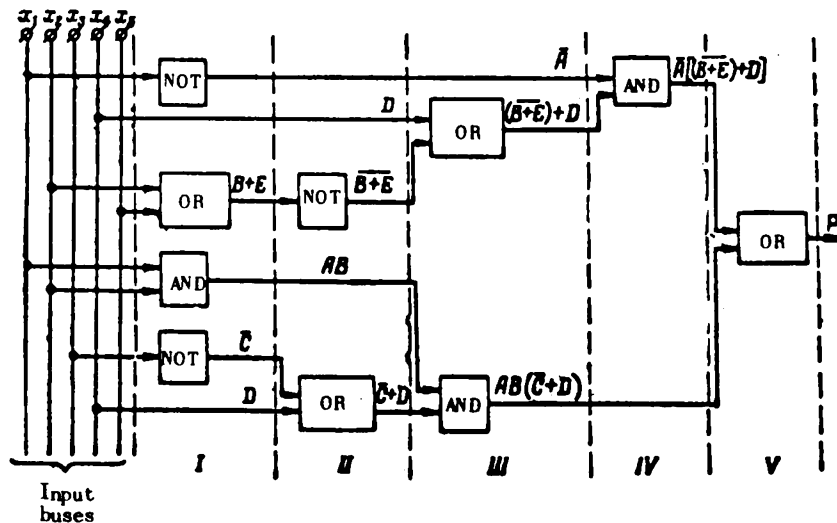


Fig.56 Function Circuit Realizing the Logic Functions (45)

Let it be required to construct the function circuit realizing the following logic function:

/122

$$P = AB(\bar{C} + D) + \bar{A}[(\bar{B} + \bar{E}) + D]. \quad (45)$$

The sequence of execution of the operations in this expression is divided into five stages.

First stage. Perform the logical multiplication  $AB$  and addition  $B + E$ ; obtain the negation of the variables  $A$  and  $C$ .

Second stage. Form  $\bar{C} + D$  and obtain the negation of the expression  $B + E$ .

Third stage. Perform the logic addition  $(B + E) + D$  and the multiplication  $AB(\bar{C} + D)$ .

Fourth stage. Perform the logic multiplication  $\bar{A}[(B + E) + D]$ .

Fifth stage. Obtain  $AB(\bar{C} + D) + \bar{A}[(B + E) + D]$ , i.e., the prescribed logic function.

In accordance with this sequence, beginning with the first stage, we construct the function circuit realizing the assigned logic function (Fig.56). The function (45) has five orders. Consequently, the maximum number of successively connected stages in the circuit through which the code signals are to pass is five.

In digital computer circuits, the input variables, i.e., the arguments of the logic function, are frequently taken from the outputs of static flip-flops. If the variable A is taken from one output of such a flip-flop, then the negation of this variable is taken from the other output. For this reason, in constructing function circuits, it may be considered that there are input buses

123

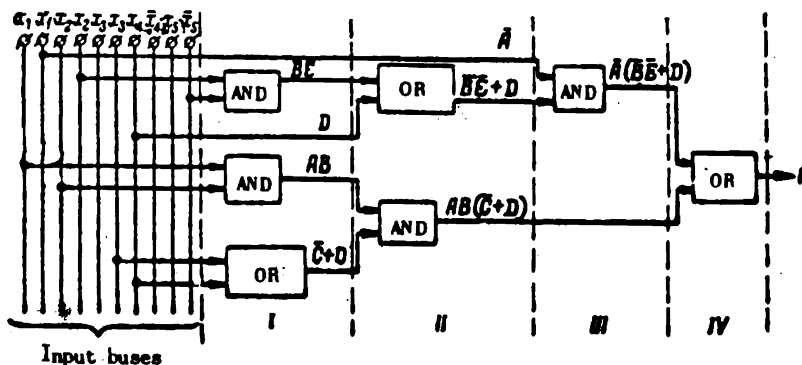


Fig.57 Function Circuit Realizing the Logic Function (46)

for all variables as well as for their negations. This permits to reduce the total number of logic elements of the circuit on account of the NOT elements which are introduced only for the case in which it is necessary to realize the operation of negation for a group of variables.

Let us illustrate this from an example. Making use of the law of inversion, we write the function (45) in the form of

$$P = AB(\bar{C} + D) + \bar{A}(\bar{B}\bar{E} + D). \quad (46)$$

Considering that there are input buses for all the variables and their negations, let us now establish the sequence of performing the operation as

given in eq.(46).

First stage. Perform the logic multiplications  $AB$  and  $\overline{BE}$  and the addition  $\overline{C} + D$ .

Second stage. Find  $\overline{BE} + D$  and  $AB(\overline{C} + D)$ .

Third stage. Find  $\overline{A}(\overline{BE} + D)$ .

Fourth stage. Find  $AB(\overline{C} + D) + \overline{A}(\overline{BE} + D)$ , i.e., the assigned function.

The function circuit realizing the logic function (46) is presented in Fig.57. The circuit is composed of seven logic elements and has four cascades. Thus, in the presence of input variables and their negations, not only will the total number of logic elements of the function circuit be lower but the number of cascades of the circuit will also decrease by one.

The construction of a function circuit for an assigned logic function can also be performed in the inverse order, in which case we must start with the formulation of the logic elements of the last cascade. For example, in realizing the function (46) the inverse order of constructing the circuit will be as follows:

1. Find  $AB(\overline{C} + D) + \overline{A}(\overline{BE} + D)$ .

2. Perform the logic multiplications  $AB(\overline{C} + D)$  and  $\overline{A}(\overline{BE} + D)$ .

/124

3. Perform the logic additions  $\overline{C} + D$  and  $\overline{BE} + D$ .

4. Find  $AB$  and  $\overline{BE}$ .

If a function circuit does not contain inverse inputs, then the total number of logic elements of the circuit is decreased by a transformation of the expression for the logic function such that it now contains the minimum possible number of negations. In this case, extensive use of the law of inversion must be made:

$$\overline{AB} = \overline{A + B},$$

$$\overline{A} + \overline{B} = \overline{AB}.$$

Realization of the left sides of these equalities is accomplished by two inverters and the AND element (OR, for the second equality) while realization of the right sides is made by one inverter and one OR circuit (AND, for the second equality).

Let us consider a few examples for laying out the function circuits of units frequently used in digital computers.

Single-digit converter. The single-digit converter  $C_0$  symbolically represented in Fig.58a has one input  $A$ , two control inputs  $S_1$  and  $S_2$ , and one out-



put P. The code signals are fed to the input A.

The purpose of a single-digit converter is to pass a signal fed to input A in direct or inverted form, depending on which of the inputs,  $S_1$  or  $S_2$ , is supplied with the control signal.

Table 13 defines the operating conditions of this converter.

TABLE 13  
OPERATING CONDITIONS OF SINGLE-DIGIT CONVERTER

A	$S_1$	$S_2$	P
1	1	0	1
0	1	0	0
1	0	1	0
0	0	1	1

The logic function of the converter, set up from the ones, is of the form

$$P = AS_1 + \bar{A}S_2$$

The function circuit realizing this expression is shown in Fig. 58b and 125 the schematic diagram in Fig. 58c.

The converter consists of a duo-triode tube acting as an inverter. The AND gates consist of diodes  $D_1$  and  $D_2$  and the OR gate of diodes  $D_3$  and  $D_4$ .

Let the high-level signal  $S_1$ , holding the diode  $D_2$  blocked, be fed to the converter. If there is no signal at the output A, the tube  $L_1$  is cut off by the negative bias voltage  $-E_c$ . The high potential of the plate of the tube  $L_1$  (point M) is transferred to the grid of tube  $L_2$ , thus unblocking it, and a low potential is formed at the point N. Since there is a low potential on the bus  $S_2$  (there is no high-level signal  $S_2$ ), a small current (by comparison with the current in the tube  $L_2$ ) flows through the resistors  $R_{a1}$ ,  $R_1$  and the diode  $D_1$ , and causes a voltage drop, via the negative pole to the output, across the resistor  $R_1$ . Thus a low potential is transferred to the output of the converter, both on the right branch through the resistor  $R_2$  and the diode  $D_4$  and on the left branch through the resistor  $R_1$  and the diode  $D_3$ . If on the input A there is imposed a high-level signal, the tube  $L_1$  is unblocked, while the tube  $L_2$  is blocked. Since the diode  $D_2$  is blocked, the high potential at point N is fed through the resistor  $R_2$  and the diode  $D_4$  to the output P. Consequently, if there is a control signal  $S_1$ , the input signal is transmitted to the output 126

of the converter in the direct (noninverted) form.

Let us now apply the high-level control signal  $S_2$  to the converter, while keeping the diode  $D_1$  cut off. There is no signal  $S_1$  so that the diode  $D_2$  is

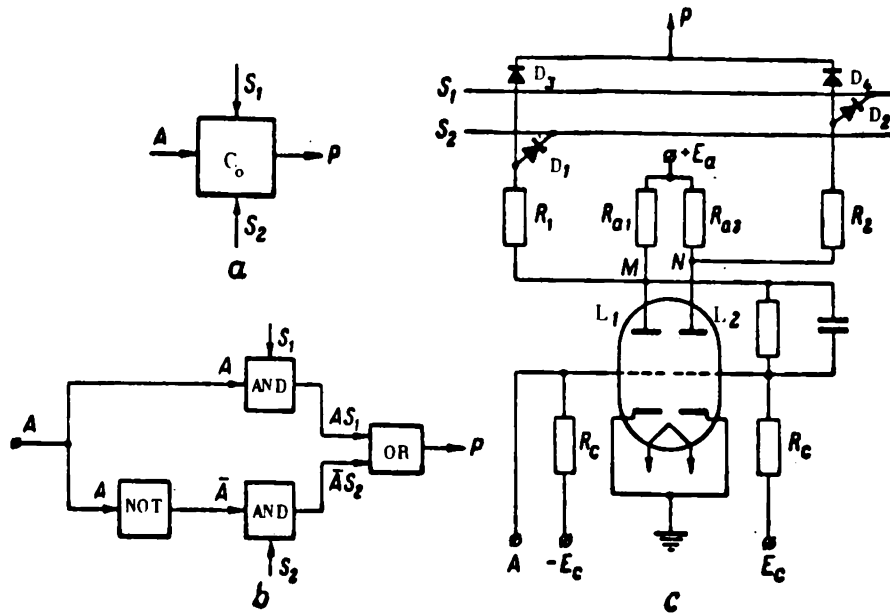


Fig.58 Single-Digit Converter  
a - Symbolic representation; b - Function circuit;  
c - Schematic diagram

open. In the absence of a positive signal at the input  $A$ , as in the previous case, the tube  $L_1$  is cut off while the tube  $L_2$  is open. Since the diode  $D_1$  is cut off, the high potential at the point  $M$  is transmitted through the resistor  $R_1$  and the diode  $D_3$  to the output of the converter. With a positive signal at the input, the state of the tubes of the triode is reversed: A low potential is formed at the point  $M$ , while a small current flows through the resistors  $R_{a1}$ , and  $R_2$ , and the diode  $D_2$ , causing a voltage drop across the resistor  $R_2$ , of polarity negative with respect to the output  $P$ . Consequently, if there is a control signal  $S_2$ , the input signal is transmitted to the converter output in inverted form.

The single-digit converter, like the nonequivalent circuit, can be used to obtain the inverse code of a binary number. In fact, if the control signal  $S_2$  is impressed on the circuit shown in Fig.58c and a binary number is then fed in code to the input  $A$ , then the inverse code of this number will appear at the output  $P$ . To obtain the inverse code of a binary number represented in parallel code, a parallel circuit of single-digit converters is necessary (as many converters in the circuit as there are digits in the binary number).

Binary switch (binary gate). Figure 59a is a symbolic representation of a

binary switch. The switch has two inputs A and B for feeding the input signals, two control inputs  $S_1$  and  $S_2$ , and one output P. /127

Operating conditions of the switch: A signal will appear at the output if

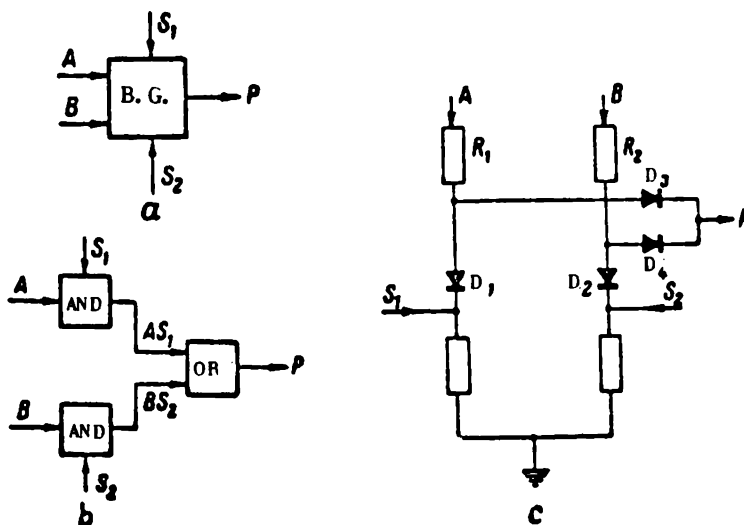


Fig. 59 Binary Switch  
a - Symbolic representation; b - Function circuit;  
c - Schematic diagram

signals are simultaneously fed to the inputs A and  $S_1$  or to the inputs B and  $S_2$  (signals are not fed simultaneously to both control inputs).

Without having recourse to a truth table, let us directly write out the logic function of the switch:

$$P = AS_1 + BS_2.$$

Figures 59b and 59c respectively give the function circuit and the schematic diagram realizing this function. The circuits are fairly simple and require no explanation.

In practice, binary switches are used for commutation (switching) of two n-digit binary numbers. If the numbers are represented in a serial code, one switch is sufficient. One number is fed to the input A and the other to the input B. Depending on the control input that receives the signal, one number or the other will be passed to the output of the circuit. If, however, the numbers are represented in a parallel code, as in single-digit converters, a parallel circuit of n binary switches is required.

## Section 23. Static Triggers

One of the most widely used elements of digital computers is the static trigger, which is a two-stage DC amplifier with a deep positive feedback. The trigger (also known as galvanic-coupled electronic relay or - better - as flip-flop circuit) was invented by the Soviet scientist M.A. Bonch-Bruyevich in 1918.

Depending on the circuit, a flip-flop may have two stable states or only one. In the former case, the flip-flop, under the action of the trigger pulse, passes from one stable state to the other, in which it remains indefinitely after the trigger pulse has stopped. The flip-flop is flopped to the initial state after the next trigger pulse. In the latter case, some definite time after the end of the trigger pulse (depending on the circuit parameters), the trigger will return to its initial state.

A flip-flop with one stable state is called a monostable multivibrator or a monovibrator. Flip-flops with two stable states or bistable multivibrators are mostly used in digital computers, for which reason we will predominantly discuss this type.

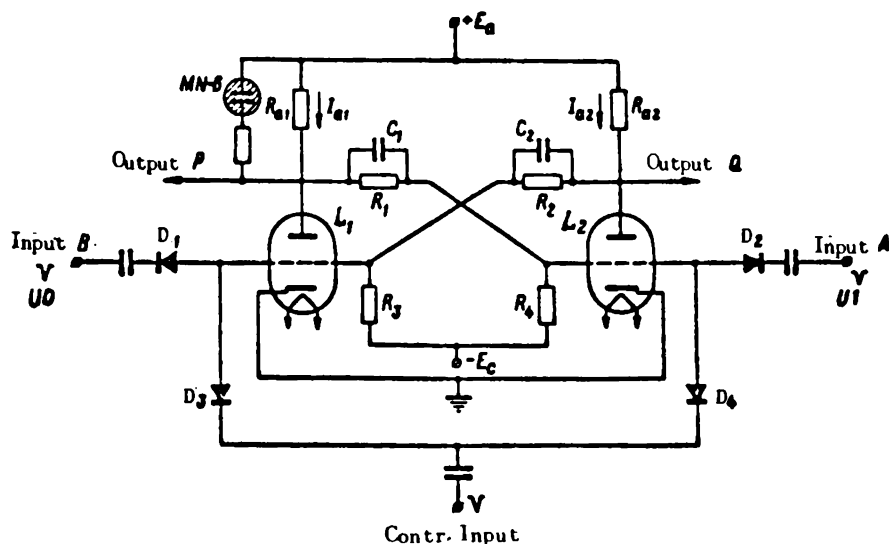


Fig.60 Static Flip-Flop using Electron Tubes

The vacuum-tube circuit for the flip-flop (Fig.60) has two electron tubes. It is characterized by mutual plate-grid coupling between the tubes and by a 128 common bias for both. The plate of the triode  $L_1$  is coupled to the grid of the triode  $L_2$  across the resistor  $R_1$  and the shunting capacitor  $C_1$ . In turn, the plate of the triode  $L_2$  is coupled to the grid of the tube  $L_1$  across the resistor  $R_2$  and the shunting capacitor  $C_2$ . The flip-flop circuit is symmetric, i.e., the tubes  $L_1$  and  $L_2$  are the same and  $R_{a1} = R_{a2}$ ;  $R_1 = R_2$ ;  $R_3 = R_4$ ;  $C_1 = C_2$ . Thus, after application of the voltage either tube may prove to be cut off. If the left tube is cut off, a high voltage will be produced on its plate, whereas

the voltage will be low on the plate of the right tube, because of the voltage drop across the plate resistor  $R_{a2}$ . Owing to the plate-grid coupling, both these voltages are transferred to the grid of the other tube. If the circuit parameters are properly selected, these grid potentials hold the tubes of the flip-flop in their original state, which will thus be stable.

Similarly, if the right tube is cut off, there will be a low voltage on the plate of the triode  $L_1$  and on the grid of the triode  $L_2$ , and a high voltage on the plate of the triode  $L_2$  and the grid of the triode  $L_1$ . This is the second stable state of the flip-flop.

The flip-flop can be switched from one stable state to the other (triggered) by feeding pulses of any polarity to the respective electrodes of the tubes. However, the transients in the electric circuit will have optimum flow if negative pulses, fed to the grids of the tubes, are used for this purpose.

In the circuit we are considering, the flip-flop is started by negative /129 pulses arriving from the inputs B and A on the grids of the triodes  $L_1$  and  $L_2$  across bypass capacitors.

Let a negative pulse be fed to the grid of the conducting triode  $L_1$ . The plate current  $I_{a1}$  of this triode will then begin to decrease and the potential on its plate rise sharply. Consequently, the positive potential on the grid of the triode  $L_2$  increases, so that the current  $I_{a2}$  appears in its plate circuit. This causes a voltage drop across the plate of the triode  $L_2$  which, in turn, leads to a further decrease of the positive grid potential and the plate current of the triode  $L_1$ , etc. As a result of this avalanche process of decreasing of the plate current  $I_{a1}$  and increasing the plate current  $I_{a2}$ , the triode  $L_1$  is cut off and the triode  $L_2$  is open, i.e., the circuit passes into a new stable state.

If a starting negative pulse is then fed to the input A, i.e., to the grid of the conducting tube  $L_2$ , then, as before, the flip-flop will pass into a state in which the triode  $L_1$  is open and the triode  $L_2$  is cut off.

The switching time (operating time) of the flip-flop, on application of a trigger pulse, is very short. This is accomplished by having the capacitors  $C_1$  and  $C_2$  connected in parallel with the resistors  $R_1$  and  $R_2$ ; these capacitors offer relatively little resistance to a pulse and shunt the high-ohmic resistors  $R_1$  and  $R_2$ . The capacitors shorten the time required to charge the stray grid-to-cathode interelectrode capacitance of the triodes. This interelectrode capacitance has an appreciable influence on the duration of the transient process; at the instant of cutoff of the tube, the positive jump of plate voltage is transferred in full to the grids of the other cut-off tube only if this capacitor is charged.

It is desirable to have the capacitance of the capacitors  $C_1$  and  $C_2$  high, in order to decrease their resistance to the passage of pulses; however, this capacitance must not be too high, since then charging and discharging of the capacitors will require excessive time, which might lead to continuous oscillation of the circuit (multivibrator regime).

A flip-flop with two stable states may have one or two inputs. In the former case, the inputs A and B are connected and trigger pulses are fed simultaneously to the grids of both tubes. Such a pulse feed is known as "feed through a counter input". In the schematic diagram, this input is denoted by co.inp. When operating as a scale-of-two counter, the flip-flop is started by its trigger pulse, and the transient process ends considerably before passage of the next pulse. If the flip-flop has two inputs, then the trigger pulse is fed only to the input A, while a pulse switching the circuit to its original state is fed to the input B.

Because of the presence of the semiconductor diodes  $D_1 - D_4$ , the trigger pulses act only on the grid of the open triode. The reason for this is the 130 fact that the diode passes current only when the trigger pulse is transferred to the grid of the open triode. As soon as the triode is cut off, the diode is also cut off, due to the low potential on the grid of the tube. Thus the diodes decouple the grid circuits of the flip-flop from the pulse sources. The

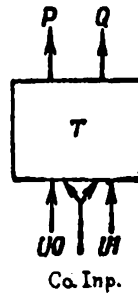


Fig.61 Symbolic Representation of the Static Flip-Flop

diodes also protect the flip-flop from duplicate operation under the action of the same negative square trigger pulse. The trailing edge of the trigger pulse which, in the absence of diodes, might flop the flip-flop, has no effect in this case.

The output signals are taken off the plates of the tubes. In order to define the stable position occupied by the flip-flop at a given instant, a neon tube (marked MN-6 on the schematic diagram) with an additional resistor, drawing only little current, is connected in parallel to one of its plate resistances. The position of the flip-flop, at which the neon will light, corresponds to the code "1" (on the plate to which the tube is connected, low potential). The opposite position of the flip-flop corresponds to the code "0".

Symmetry of the loads connected to the plates is essential to the operation of the flip-flop. In many cases, the plates from which the output signals are taken are therefore not coupled with the load directly but across cathode followers (for example in the Soviet computers M-2 and M-3). This eliminates the influence of the load on the flip-flop circuit.

Figure 61 gives the symbolic representation of the static trigger used in function circuits. The trigger pulse, fed to the input U0 sets the flip-flop in the position of the code "1", while the pulse fed to the input U1 resets it to the code position "0". We will retain these symbols for the flip-flop hereafter, regardless of the elements of which the flip-flop are constructed.

In pulse circuits, especially in those with two stable states, semiconductor triodes are effectively used, since here they operate in the "on-off" regime and the stability of their characteristics need not be very high.

When point-contact semiconductor triodes are used, the flip-flop circuit needs only a single triode. This is the main advantage of point-contact triodes over junction triodes. They are almost equivalent in economy, however, since a considerable number of semiconductor diodes must be connected to assure stability of operation of flip-flops using point contact triodes. At the same time, flip-flops using junction triodes have more stable characteristics, are more reliable, and have a lower resistance in the nonconducting state. Flip-flops with junction triodes have, therefore, found wider application. /131

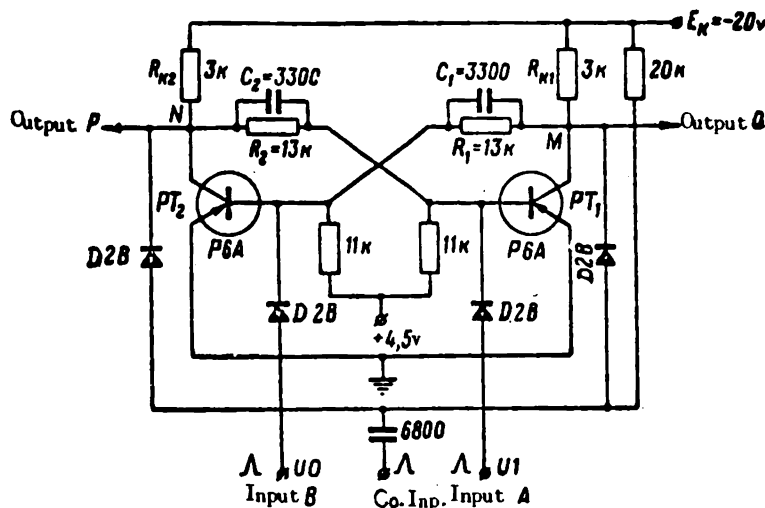


Fig.62 Static Flip-Flop using Semiconductors

A flip-flop circuit with two junction triodes resembles the analogous vacuum-tube circuit. Figure 62 gives a flip-flop circuit using two p-n-p junction triodes. The trigger pulses of positive polarity are fed to the base of the triodes, causing their cutoff.

If a positive voltage pulse is fed to the input A, then the triode PT<sub>1</sub> is cut off and the collector current passing through the resistor R<sub>1</sub>, will decrease to a negligible value. In this case, a low potential is formed at the output Q (at the point M), since the voltage on the collector triode PT<sub>1</sub> varies from zero to a value close to E<sub>N</sub>. Owing to the presence of feedback, this low potential is transmitted through the filter R<sub>1</sub>C<sub>1</sub> to the base of the triode PT<sub>2</sub>,

which becomes conducting. A relatively large collector current begins to flow through the resistor  $R_{k2}$ , while the voltage on the collector of the triode  $PT_2$  varies from  $E_k$  to almost zero. Consequently, on the output P (at the point N) a relatively high potential arises. The flip-flop remains in this position until a positive pulse appears at the input B and flops the flip-flop to its original state. The capacitors  $C_1$  and  $C_2$ , as in a vacuum-tube flip-flop, serve to accelerate the transient processes. In this circuit, the neon for visual indication of the state of the flip-flop cannot be directly connected in the collector circuit of the triode, since the collector voltage is insufficient to light it. The output signals are therefore fed to an amplifier consisting of a semiconductor triode with a higher collector voltage sufficient to light the neon connected in its collector circuit. /132

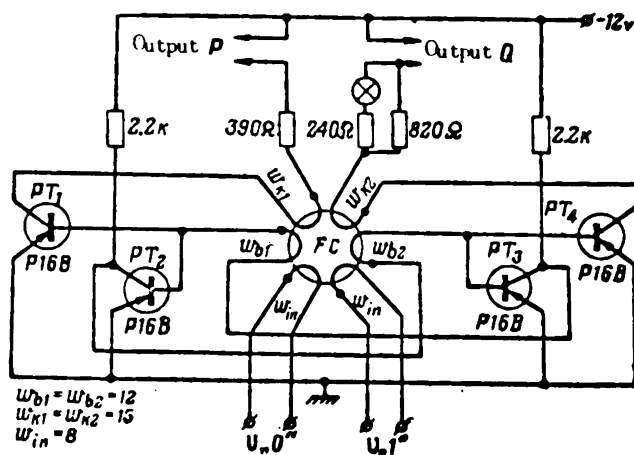


Fig.63 Static Ferrite-Transistor Flip-Flop

In units consisting of ferrite-transistor elements, it is convenient to use a static ferrite-transistor flip-flop (Fig.63) based on a flip-flop circuit of direct-coupled semiconductor triodes. In contrast to the direct-coupled flip-flop, a static ferrite-transistor flip-flop circuit uses a memory transformer, ensuring a more reliable determination of the state of the flip-flop. A ferrite core with rectangular hysteresis loop is used as the transformer core and decreases the sensitivity of the flip-flop to noise from the inputs.

The ferrite-transistor flip-flop is composed of the semiconductor triodes  $PT_2$  and  $PT_3$  and the transformer. These triodes function as amplifiers. Use of the output amplifiers makes it possible to increase the output current of the flip-flop to 30 ma and to decouple the load. Since the resistance of the transformer windings is low, the DC coupling in the ferrite-transistor flip-flop is practically the same as in the direct-coupled flip-flop. Therefore, the high resistance of the flip-flop to excitation at high temperatures is maintained. Such a flip-flop operates reliably under temperature variations from  $-60$  to  $+80^\circ\text{C}$  at frequencies up to 300 kc and is insensitive to noise in the input and feed circuits. Noise of amplitude less than 30% of the amplitude of the input pulses (the rated amplitude of these pulses is 150 ma) will not operate the



flip-flop.

Another favorable feature of this flip-flop is the fact that the triodes need not be selected for symmetry of characteristics. To ensure reliable saturation of the output amplifiers of the flip-flop, the input impedance of the triodes  $PT_1$  and  $PT_4$  must not exceed the input impedances of the base-coupled triodes  $PT_2$  and  $PT_3$ . /133

The static flip-flop is used in digital computers as memory register, frequency divider, counter element, and binary switch.

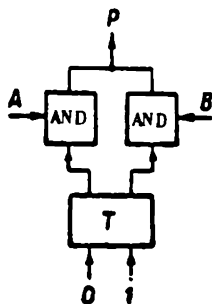


Fig.64 Binary Switch using a Static Flip-Flop

The use of a flip-flop as a memory register is based on its property of maintaining one of its stable states for an indefinite time. In this case, the inputs of the flip-flop are not connected and are used as follows: one for data writing (1 or 0), i.e., feeding the pulse that sets the flip-flop into the required state, and the other for reading out the stored information, i.e., feeding the pulse that resets the flip-flop to the initial state.

The flip-flop is used as a frequency divider to halve the pulse repetition rate. The inputs of the flip-flop in this case are connected, i.e., the pulses are fed through the counter input. A pulse of definite polarity on one of the flip-flop outputs arises when every other pulse of a certain series arrives at the counter input.

Figure 64 shows a circuit in which the flip-flop acts as a binary switch. The flip-flop controls two logic circuits (AND gates). Depending on the position of the flip-flop, either the signal A or the signal B is passed to the output P.

The use of a flip-flop as a counter element will be discussed below.

A monostable multivibrator, or a flip-flop with one stable state, is used in circuits requiring a cell that can independently return to its initial position a certain time after switch-over. This device is used to delay pulses for a certain time, determined by the parameters of the constituent elements. Such a one-shot multivibrator is used instead of a bistable multivibrator for switch-

ing circuits, when the switch must return to the initial position a definite time after the arrival of a control signal. A "monovibrator" is also used to shape broad pulses into the narrow trigger pulses fed to its input.

Figure 65a is a schematic diagram of the circuit of a one-shot multivibrator with semiconductor triodes. In the stable state of the circuit, the triode  $PT_2$  whose base is directly connected across the resistor  $R_1$  with the negative pole of the feed source is conducting while the triode  $PT_1$  is cut off. When the negative voltage pulse  $u_{in}$ , fed to the base of the triode  $PT_1$ , arrives at the input of the circuit, the flip-flop passes into the unstable state. /134 The triode  $PT_1$  begins to conduct current, and the voltage variation on its

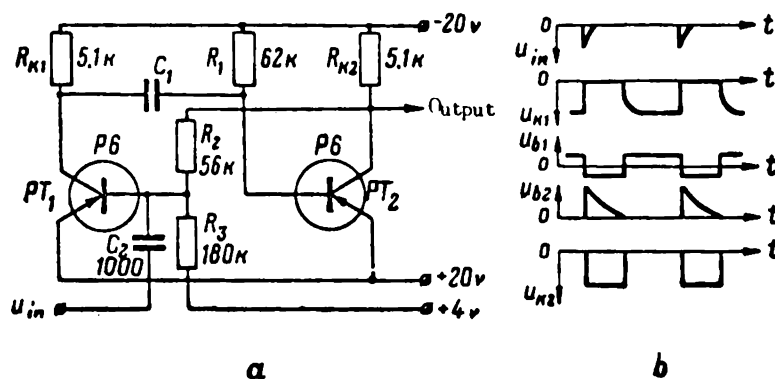


Fig.65 Monostable Multivibrator  
a - Schematic diagram; b - Shape of pulses on the triode electrodes

collector is transmitted across the capacitor  $C_1$  to the base of the triode  $PT_2$ . The increasing avalanche process, which takes place on account of the feedback (collector of the triode  $PT_2$  to base of the triode  $PT_1$ ) leads to complete cut-off of the triode  $PT_2$ , while the triode  $PT_1$  is completely open. The flip-flop will remain in this state until the capacitor  $C_1$  discharges across the collector of the conducting triode  $PT_1$ , the resistor  $R_1$ , and the input resistor of the triode  $PT_2$ .

After discharge of the capacitor  $C_1$ , a negative voltage is established on the base of the triode  $PT_2$ , and as a result this triode is placed in the conducting state while the triode  $PT_1$  is cut off due to the presence of feedback, and the flip-flop returns to its initial position.

Thus, a pulse of negative polarity with steep fronts occurs at the output of the flip-flop. Figure 65b shows the shape of the pulses on the triode electrodes.

## Section 24. Dynamic Flip-Flops

A flip-flop in which the signal at the output, corresponding to the state 1, is represented by the presence of pulses of definite frequency, and the state 0 by the absence of pulses is known as a dynamic flip-flop. Thus, in a dynamic flip-flop, a code signal 1 or 0 fed to the input is maintained in the dynamic form, i.e., in the form of the presence or absence of a train of pulses at the output.

A dynamic flip-flop, whose symbolic representation is shown in Fig.66a, has three inputs and one output. Sync pulses (clock or timing pulses), of 135 definite frequency are continuously fed to the input SP. When the code signal 1 is fed to the input U1, the flip-flop is set in position 1, and a pulse train, at the repetition rate of the sync pulses, appears at the output. As soon as a pulse is fed to the input U0, the generation of pulses at the output stops.

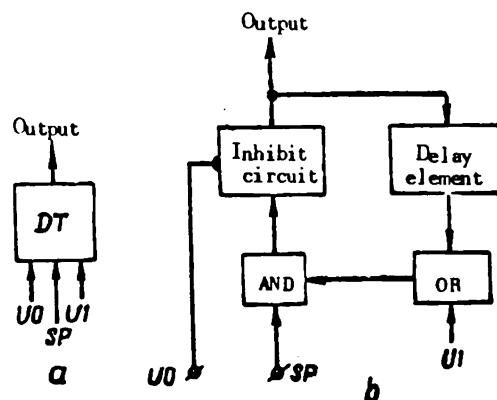


Fig.66 Dynamic Flip-Flop  
a - Symbolic representation; b - Function circuit

The principle of operation of a dynamic flip-flop is that a pulse arriving at the input U1 will circulate in the flip-flop, constantly regenerating its shape and duration and being fed at definite time intervals to the output of the circuit. Actually, the code signal 1 fed to the input U1 passes through the OR element (Fig.66b) and then, simultaneously with one of the sync pulses, arrives at the coincidence circuit. From the output of the AND element, the pulse arrives at the inhibit circuit and is passed by it, since the signal  $U^0$  is absent from the left input of the inhibit circuit. From the output of the inhibit circuit, the pulse arrives at the delay element, whose time delay equals the pulse repetition rate of the sync pulses. Artificial delay lines or capacitors are ordinarily used as delay element. The output signal from the delay element, passing through the OR gate, arrives at the coincidence circuit at the instant of the regular sync pulse action. From the output of the AND element, the pulse again arrives at the inhibit circuit, and so on. This ensures circulation of a pulse around the closed circuit, giving pulses at the repetition rate of the sync pulses at the flip-flop output.

The flip-flop is reset to position 0 by feeding a pulse  $U''0''$  to the inhibit input of the inhibit circuit at the instant when the signal from the AND element reaches the other input. In this case, the chain of circulation is broken, since the signal from the AND gate does not pass to the inhibit circuit. In that case, the code "0" is remembered in the flip-flop until a new code signal 1 arrives at the input  $U1$ , which repeats the circulation of the pulses and feeds them through the flip-flop output. A shaping element must be included in the circulation loop to restore the shape and amplitude of the pulses fed to the input of the delay element. These functions may be performed by the AND element or by the inhibit circuit. /136

Dynamic flip-flops using delay lines as the delay element are rarely used in Soviet digital computers, since they have the following drawbacks:

a) The code signals  $U''1''$  and  $U''0''$  as well as the sync pulses, must be strictly matched in time.

b) All elements of the circulation loop, together with the delay line, must provide a delay time equal to the repetition rate of the sync pulses.

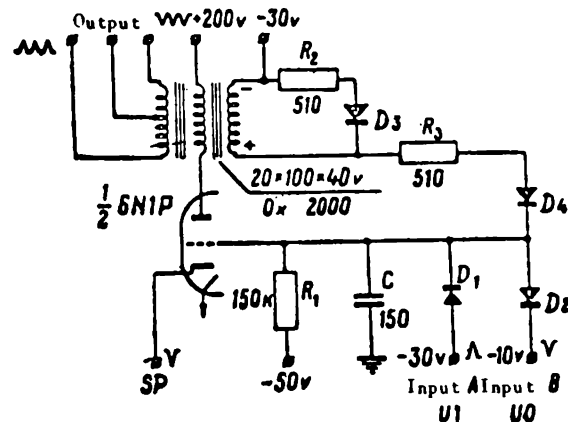


Fig.67 Schematic Diagram of Dynamic Flip-Flop with Capacitive Memory

c) To secure reliable operation of the coincidence circuit when the frequency of the sync pulses or the delay time of the circulation loop vary, the duration of the pulses arriving from the delay line through the OR circuit at the AND element must be somewhat longer than the duration of the sync pulses.

The dynamic flip-flop developed by Soviet designers, using a capacitor as the delay element, is more widely used. This device is known as a dynamic flip-flop with capacitive memory.

The operation of a dynamic flip-flop with the memory capacitance is based on the rapid charging of a capacitor across a small resistance and its slow

discharge across a larger resistance. Figure 67 shows one of the practical versions of the circuit of a dynamic flip-flop with capacitative memory, using a 6N1P vacuum tube. Here, as in the flip-flop with delay line, a continuous pulse train appears at the output and coincides in time with the sync pulses if the circuit is set in the code position "1".

In the absence of the code pulse "1", the tube is cut off and the sync pulses arising at its cathode are not passed to the output of the circuit. A code pulse "1" fed through the diode  $D_1$  to the input A, charges the capacitor C, and opens the tube for the sync pulses. The charging of the capacitor is of very short duration, since the DC resistance of the diode is small. After 137 passage of the first sync pulse, pulses arise in the secondaries of the transformer; these pulses pass from the left secondary winding to the output of the circuit, and from the right secondary winding to the feedback circuit to restore the capacitor charge. The capacitor is connected with the large resistor  $R_1$ , so that it has no time to discharge in the intervals between the pulses SP. Consequently, the tube is always conducting for the sync pulses. Thus, it is sufficient to feed to the capacitor C a single positive pulse corresponding to the code "1", in order to have pulses following at the repetition rate of the sync pulses appear at the flip-flop output. Pulse generation at the output continues until a pulse of negative polarity corresponding to the code "0" appears at the input B; at this time, the capacitor C rapidly discharges across the conducting diode  $D_2$ , and the tube is cut off.

The resistor  $R_2$  and the diode  $D_3$ , connected in parallel with the right secondary winding of the transformer, serve to quench the parasitic oscillations arising in the transformer. The resistor  $R_3$  in the feedback circuit is used to weaken the coupling.

The flip-flop has the following rating:  $u_a = 200$  v;  $u_{tr.u.} u_1 = -30$  v;  $u_{tr.u.} u_0 = -10$  v;  $u_{R_1} = -50$  v; transformer with oxyfer core (oxyfer 2000); core dimensions: outside diameter 7 mm, inside diameter 4 mm, thickness 3 mm; the primary of the transformer has 100 turns and the secondaries have 20 and 40 turns; the diodes are of type DGTs-4, -5, -6; their frequency range is within 100 - 1350 kc; the sync pulses are of bell shape with an amplitude of 20 v and a duration of 0.4 - 0.5  $\mu$ sec in the base; the pulses U"1" and U"0" are of bell shape, of amplitude 25 v and duration 0.4 - 0.5  $\mu$ sec in the base.

A comparison of the dynamic flip-flop with the capacitative memory and the vacuum-tube circuit of the static flip-flop indicates the following advantages of the former over the latter:

1. A dynamic flip-flop has a higher operating speed; in static flip-flops, this speed is limited by the transient processes.
2. The transformer output of the dynamic flip-flop yields pulses of different polarity and, if necessary, permits matching with units using semiconductor devices with a low input impedance.
3. A dynamic flip-flop uses less power, because of the fact that, in the state 0, the vacuum tube of the flip-flop is cut off and its plate current is

zero. In position 1, the pulses of the plate current occur only at the instant of arrival of the sync pulses. In the static flip-flop, one of the tubes is always conducting and consumes energy.

Figure 68 shows the circuit of the dynamic flip-flop with capacitive memory, developed at the Institute of Fine Mechanics and Computer Technology. High-frequency semiconductor triodes P402-P403 are used. The basic element <sup>/138</sup> of the circuit is the pulse-potential gate. The capacitive memory C is connected through an emitter follower to one input of this gate, while the sync pulses SP are continuously fed to the other input.

If the flip-flop is in position 0, then the capacitor C is discharged, the potential at the output of the emitter follower is close to 0, and the diode D<sub>5</sub> is cut off. Owing to this fact, the entire collector current appearing at the

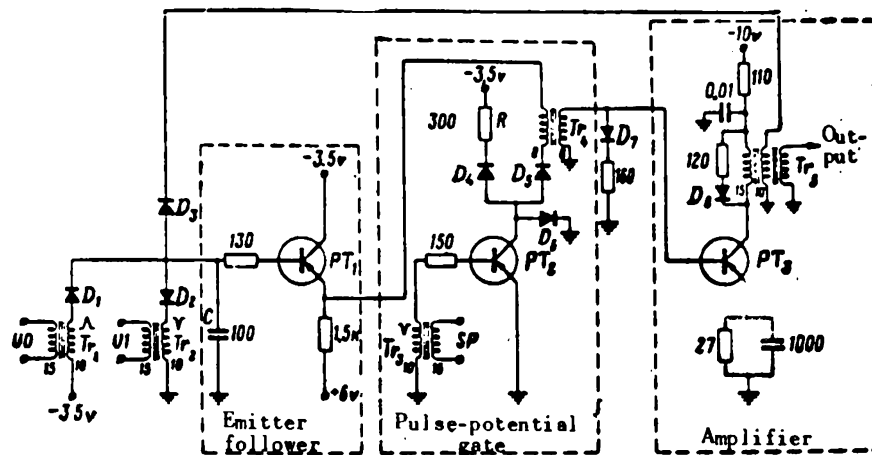


Fig.68 Schematic Diagram of the Semiconductor-Triode Dynamic Flip-Flop

instant at which the sync pulses are fed to the base of the triode T<sub>2</sub>, flows through the diode D<sub>4</sub> and the resistor R. There are no pulses at the output of the flip-flop.

To set the flip-flop in position 1, the negative pulse U"1" is fed through the diode D<sub>2</sub> and charges the capacitor C to -3.5 v. If the amplitude of the pulse U"1" is greater than the rated value, the capacitor will discharge to -3.5 v across the diode D<sub>1</sub>. The diode D<sub>5</sub> then is conducting. The sync pulses to be fed to the input of the pulse-potential gate can now pass through either the left collector filter D<sub>4</sub>/R or through the right collector filter D<sub>6</sub>/Tr<sub>4</sub>. After amplification, these pulses respectively arrive at the flip-flop output or flow through the feedback circuit and the diode D<sub>3</sub>, to restore the charge of the capacitor C. The flip-flop remains in position 1 until a positive U"0" discharges the capacitor. This flip-flop circuit operates at a frequency of several megacycles.

The dynamic flip-flop with capacitative memory has a number of advantages over the flip-flop with delay line:

1. The entire unit is smaller in size owing to the absence of a delay line, which latter is one of the most bulky elements of the circuit. /139
2. The permissible frequency variation range of the sync pulses is many times larger than in delay-line flip-flops. Even a skipping of several sync pulses or a sharp variation in their amplitude will not dump the circuit, whereas skipping of a pulse or a sharp amplitude drop of even a single sync pulse in a flip-flop circuit with delay line will stop the feeding of pulses to the output.
3. The flip-flop with capacitative memory does not require strict synchronization of the input signals.

The disadvantages of the dynamic flip-flop with capacitative memory include: necessity of a large number of different feed sources, difficulty of adjustment, and impossibility of representing a flip-flop state (1 or 0) in paraphase code.

In special-purpose digital computers with ferrite-transistor cells, dynamic ferrite-transistor flip-flops are used. These are also known as generating cells or generators.

A generating cell (Fig.69) is built of three ferrite-transistor cells with push-pull feed in the case under discussion. Let all the cells of the dynamic flip-flop be in position 0. A code signal fed to the input U1 sets the ferrite core FC<sub>1</sub> into position 1. Then the read pulse  $i_{r,1}$  returns the core to position 0, while the triode PT<sub>1</sub> becomes conducting and the collector current  $i_{k,1}$ , flowing through the winding  $w_{r,3}$  switches the core FC<sub>3</sub> to position 1. The pulse  $i_{k,1}$  is the output pulse of the flip-flop. Then the pulse  $i_{r,2}$  resets the core FC<sub>3</sub> into its initial state 0, and the triode PT<sub>3</sub> opens causing the current pulse  $i_{k,3}$  to appear. The current  $i_{k,3}$ , passing through the feedback circuit, arrives at the winding  $w_{r,1}$  thus producing a second switching of the core FC<sub>1</sub> into position 1. The regular read pulse  $i_{r,1}$  again returns the core FC<sub>1</sub> to position 0, and the current  $i_{k,1}$  sets the core FC<sub>3</sub> into position 1, and so forth. Thus, presence of positive feedback between the third and first ferrite-transistor cells ensures continuous circulation of the code signal 1 in the flip-flop circuit, as well as the feeding of pulses spaced at the read-pulse repetition rate to the flip-flop output (see flow chart of the flip-flop in Fig.69b).

The circulation of the code signal 1 in the closed loop consisting of the first and third cells is interrupted, i.e., the flip-flop is switched to the position 0 by a pulse fed to the input U0. In fact, on arrival of this pulse, the core FC<sub>2</sub> is switched into position 1. Then the regular read pulse  $i_{r,1}$  /140 returns it to position 0. Here the triode PT<sub>2</sub> is completely conducting, since there is an independent circuit for its collector current. The current  $i_{k,2}$  causes a voltage drop across the resistor R<sub>1</sub>, thus suppressing the emf that arises in the base winding of the core FC<sub>1</sub> when it is switched by the pulse  $i_{r,1}$  into position 0. For this reason, at the instant at which the core FC<sub>1</sub> is

switched to position 0, the triode  $PT_1$  remains cut off and no current pulse  $i_{k1}$  appears in the write winding  $w_{w3}$  of the core  $FC_3$  nor at the circuit output. Consequently, the regular read pulse  $i_{r2}$  will not change the state of magnetization of the core  $FC_3$ , the circulation of the pulse in the feedback loop will

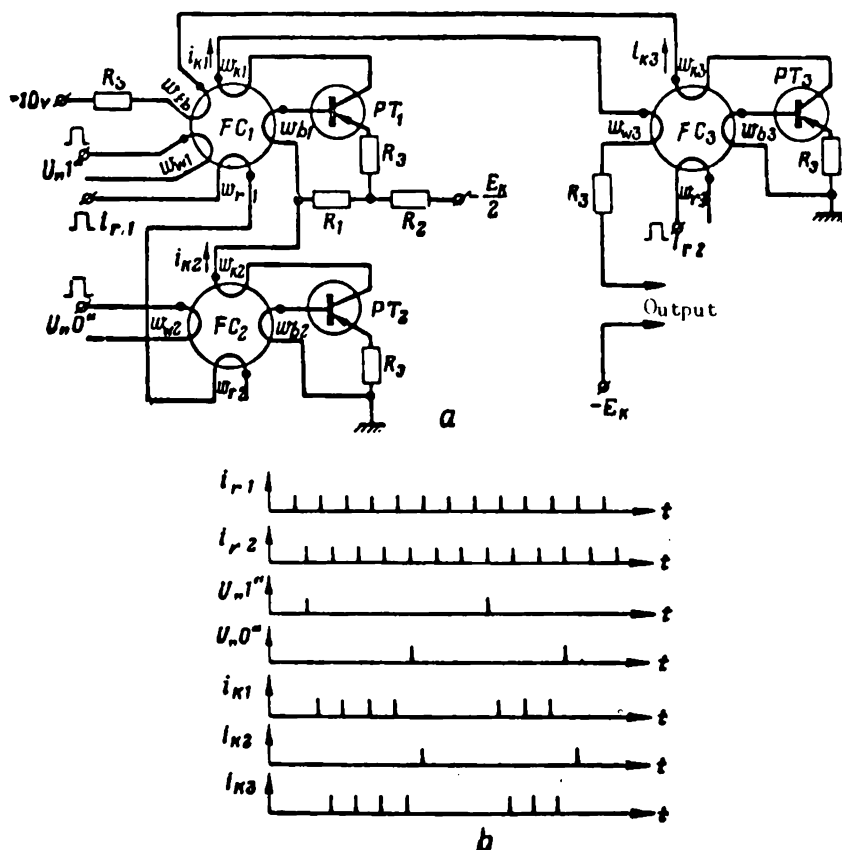


Fig.69 Dynamic Flip-Flop of Ferrite-Transistor Cells  
(Generating Cells):  
a - Schematic diagram; b - Flow chart of operation

stop, and the flip-flop will be set into position 0.

141

Ferrite-transistor cells are ordinarily used to construct a dynamic flip-flop. In contrast to the others, the core  $FC_1$  here has two write windings  $w_{w1}$  and  $w_{w2}$ , so that the first cell is an OR element with two inputs. The first and second cell are from an inhibiting circuit as described above (Fig.39). Here, as in the case of logic elements made of ferrite-transistor cells (Figs.39 and 41), the readout ampere-turns must be sufficient to switch the cores without the additional magnetic field formed on account of the presence of positive feedback in the cell circuit.

A dynamic flip-flop with ferrite-transistor cells can also be constructed



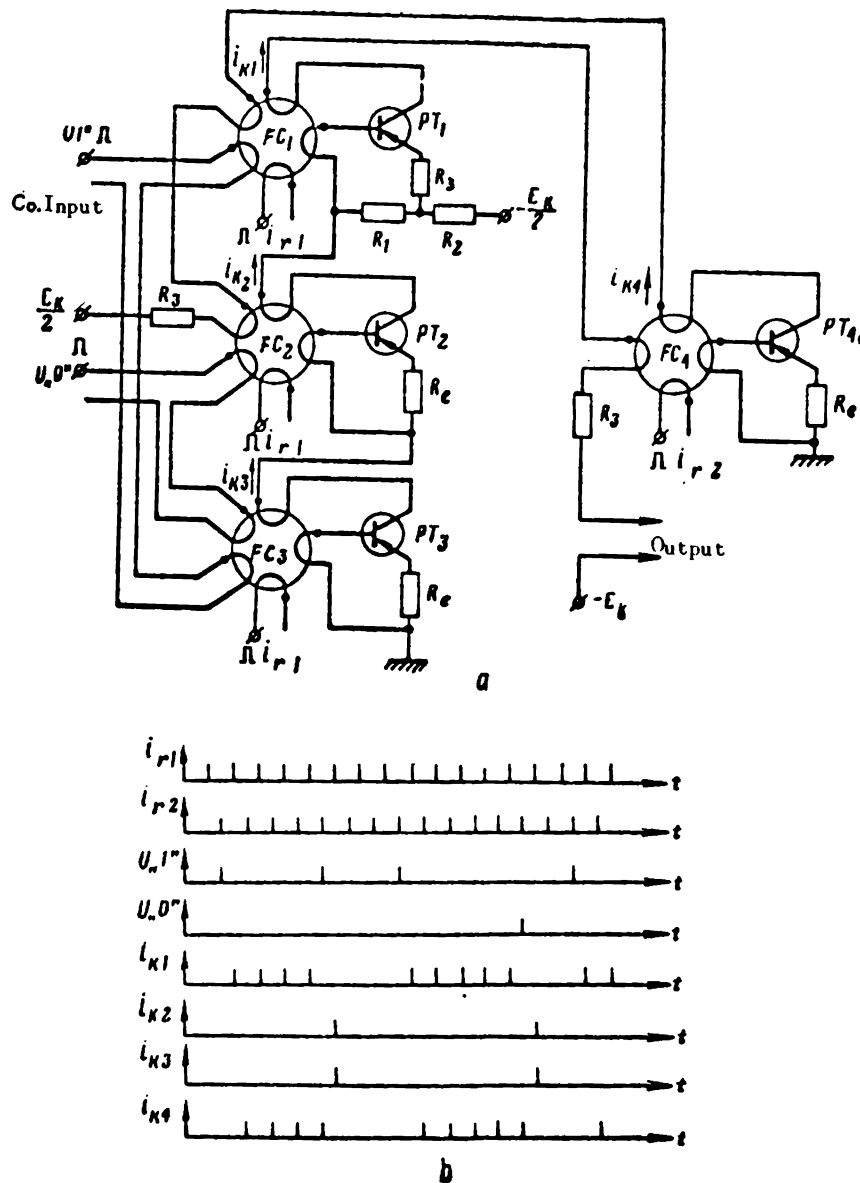


Fig.70 Dynamic Flip-Flop with Counter Input from Ferrite-Transistor Cells  
a - Schematic diagram; b - Flow chart of operation

with a counter input. In this case, the flip-flop is made of four cells (Fig.70). Each cell of the first stage is an OR logic element with two inputs. The first and second cells are connected to form an inhibiting circuit, while the second and third cells form a coincidence circuit.

The flip-flop operates as follows (see the flow chart in Fig.70b). On arrival of the first pulse of positive polarity at the counter input, the core  $FC_1$  is set in position 1; then, with regular feeding of the read pulses  $i_{r,1}$  and  $i_{r,2}$ , the code pulse 1 will circulate in the closed loop formed by the first and fourth cells. Here a train of current pulses  $i_{k,1}$  appears at the output. The regular switching of the core  $FC_3$  is not accompanied by opening and cutoff of the triode  $PT_2$ , since the circuit for the collector current  $i_{k,2}$  is open. On arrival of the second pulse at the counter input, simultaneously with the current pulse  $i_{k,4}$  at the feedback circuit, all cores of the first stage cells are set in position 1. The regular read pulse  $i_{r,1}$  resets these cores to position 0, while the triodes  $PT_2$  and  $PT_3$  start conducting and the triode  $PT_1$  remains in the cut-off position, since the first and second cells are connected in an inhibit circuit. Consequently, circulation of the pulse in the closed loop is interrupted and the flip-flop is set into position 0. It is not difficult to prove that the third pulse fed to the input U1 will again set the flip-flop into position 1, the fourth pulse into position 0, and so on.

In addition to the counter input, the flip-flop has the input U0. To this input the pulse is fed simultaneously with the arrival of the current pulse  $i_{k,4}$  in the feedback windings of the first and second cells. The pulse, arriving at the input U0, flops the flip-flop to position 0.

Dynamic flip-flops are used in digital computers to build counters, registers, control units, arithmetic units, etc.

## Section 25. Binary Counters

/143

Counting in digital computers is a process of determining the number of pulse-type signals appearing successively on one line. Pulse counters are widely used in control units of digital computers (to count the instruction numbers, to count the cycles in the operations of multiplication and division) in devices for the conversion of continuous quantities into a numerical code, in digital servosystems, etc.

Depending on the system of notation adopted in a counting unit, we distinguish binary, ternary, decimal, and other counters. Binary counters are economically most advantageous.

In accordance with the design principle, pulse counters are divided into two main groups, direct and stepping counters. The counters of the first group use an accumulative capacitor which is charged by the pulses to be counted, and the voltage on the capacitor is proportional to the number of pulses fed to it. Such a counter is essentially a converter of a discrete quantity - the number of pulses - into a continuously variable quantity - the voltage on the capacitor. The voltage is measured by a sensitive pointer instrument, with the



Thus, the flip-flop is a modulo 2 counter. By connecting in series  $n$  similar flip-flop elements, an  $n$ -digit binary counter, which counts the number of pulses fed to its input, will be obtained.

Add and subtract counters. Let us consider the circuit of a four-digit binary step counter, built up of static flip-flops (Fig.71). Each elementary register is constructed according to the circuits shown in Fig.62. The pulses of positive polarity to be counted are fed to the input of the flip-flop  $T_1$ . The input of each successive flip-flop is coupled across a capacitor to the collector of the right-hand triode of the preceding flip-flop.

Let all flip-flops in the initial state be in the code position "0". After the first input pulse is fed, the first flip-flop passes into state 1 and a low potential appears at its right-hand output, while a negative pulse appears at the input of the flip-flop  $T_2$ . Since only a positive pulse can change the state of a given flip-flop, the second, third, and fourth flip-flops remain in the initial state. The second pulse fed to the counter input sets the flip-flop  $T_1$  into the state 0; from its output, a positive pulse is transmitted to the input of the second flip-flop, setting the flip-flop  $T_2$  into the state 1. The negative pulse appearing at the input of the third flip-flop does not change its state. The third pulse fed to the counter input again switches the first flip-flop into the code position "1"; at the input of the second flip-flop, a negative pulse appears but does not change its state. Consequently, after a third pulse, the first and second flip-flops are in the code position "1" and the third and fourth in the code position "0". On arrival of a fourth pulse at the counter input, the first flip-flop is switched to the state 0; the positive pulse that now arises at its output sets the flip-flop  $T_2$  also into the state 0. Finally, a positive pulse from the output of the second flip-flop switches the third flip-flop from the state 0 into the state 1. Thus, the second flip-flop passes into a different stable state after every two pulses fed to the counter input. By similar reasoning, it can be proved that the third flip-flop will switch to a different state after every four pulses, the fourth after eight pulses, and so on. If the counter contains  $n$  flip-flop elements connected in series, then only after the passage of  $2^n$  input pulses will a single pulse appear at the counter output.

A binary counter consisting of flip-flops connected in series obeys the following law: Whenever the preceding flip-flop switches from the state 1 to the state 0, the following flip-flop is switched to the next state (i.e., from 0 to 1 or from 1 to 0). This means that, after application to the counter input, for instance, of nine pulses and after termination of the transient processes, the flip-flops will be in the following states: the fourth in state 1, the third in state 0, the second in state 0, the first in state 1. Since the first flip-flop represents the first digit of a binary number, the second flip-flop the second digit, etc., these states of the counter flip-flops thus correspond to the binary number 1001, or the decimal number 9.

To return the counter flip-flops to their original state, a wide positive pulse is fed to the common bus connected to the bases of the left-hand triodes of all flip-flops.

The maximum repetition rate of the pulses to be counted is determined by the switching time of the first flip-flop in the common circuit of the counter, since this flip-flop trips several times as often as the others. For this reason, to increase the speed of the counter, the speed of the first flip-flop must be increased. Obviously, if the interval between two pulses is shorter than the switching time of the first flip-flop, the second pulse will not be registered by the counter, and there will be a so-called miss of the circuit, or the loss of a pulse in counting.

The described counter operates only in the forward direction, i.e., in 146 the direction of increase (addition of pulses). If, however, the flip-flops of the counter are connected as shown in Fig.72 (the counter input of each successive flip-flop being coupled to the collector of the left-hand triode of the preceding flip-flop), then inverse counting (subtraction of pulses) can be performed.

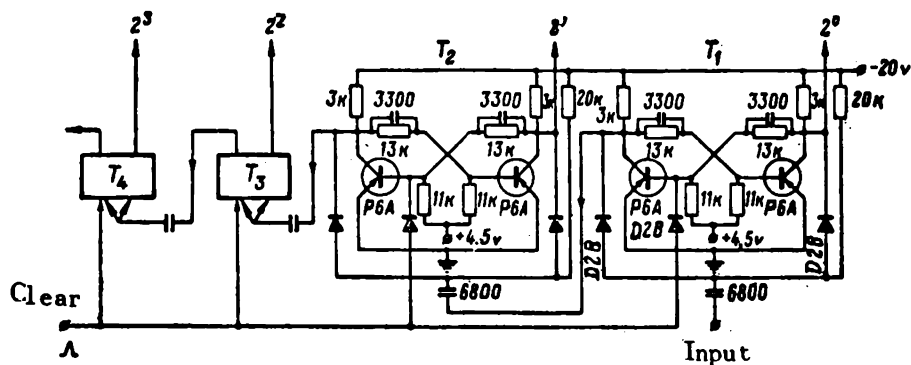


Fig.72 Four-Digit Binary Counter Operating in the Direction of Subtraction of Pulses

The operation of a binary counter in the inverse direction (subtract counter) proceeds as follows: Let the reading of the counter be three, i.e., let the first and second flip-flops be in the state 1 and the third and fourth in the state 0. When the first positive pulse is fed to the counter input, the first flip-flop is switched to the state 0. The negative pulse arising at the input of the second flip-flop does not change its state. Consequently, the reading of the counter has now become two, because the second flip-flop, in the state 1, represents the second digit of the binary number. On application of a second pulse to the counter input, the first flip-flop is switched to the state 1, and the second to the state 0, since a positive pulse appears at its input. The counter now shows one. After the third input pulse, the first flip-flop switches to the state 0, while the state of the other flip-flops does not change, and the counter reading is now zero. A fourth pulse switches all flip-flops of the counter to the state 1; the reading of a counter with four flip-flop elements equals the binary number 1111, or the decimal number 15. Beginning with this number, on application of pulses to the input, the counter will operate in the inverse direction.

Table 14 shows the sequence of operation of a binary counter in the inverse direction. In the initial position, i.e., before arrival of the first pulse at the counter input, all the counter flip-flops are in position 1.

TABLE 14

147

SEQUENCE OF OPERATION OF COUNTER IN INVERSE DIRECTION

Flip-Flops Input Pulses	$T_4$	$T_3$	$T_2$	$T_1$
1	1	1	1	0
2	1	1	0	1
3	1	1	0	0
4	1	0	1	1
5	1	0	1	0
6	1	0	0	1
...	...	...	...	...
...	...	...	...	...
...	...	...	...	...
14	0	0	0	1
15	0	0	0	0

The counting rate of the discussed add-subtract counters is limited by the maximum operating speed of the first flip-flop. The speed of counters also depends on the transfer time of carry ones along the counter loop. In the worst case, when the carry one is carried from the least significant digit to the most significant, the setup time of an  $n$ -digit counter is  $n\tau$ , where  $\tau$  is the switching time of a flip-flop. Obviously the readings of a counter in parallel code can be taken only after it has been set, i.e., only after the transient processes have ended in all flip-flops.

The speed of a flip-flop counter can be materially increased if the carry of ones along the counter loop is not successive from the lowest place to the highest place through all the intermediate places, but is simultaneous in all places. Flip-flop counters with simultaneous carry of ones for all places are sometimes called continuous-carry counters.

Continuous carry counter. To provide simultaneous carry of ones in all digit positions, an add counter is provided with auxiliary elements. In the circuit of Fig. 73, pulse-potential gates are used as such elements.

The concept of a counter with continuous carry is based on the following peculiarity of addition of binary numbers: If a unit of the least significant digit is added to a certain binary number, then the sum can be obtained by 148

replacing the first zero (counting from right to left) by a one, and all ones to the right of that zero by zeros. For example:

$$\begin{array}{r} + 101010111 \\ 1 \\ \hline 101011000 \end{array}$$

In the first summand, the first zero on the right is in the fourth digit-position. Replacing this zero by a one, and the ones in the lower-order positions by zeros, we obtain a sum greater by one than the first summand.

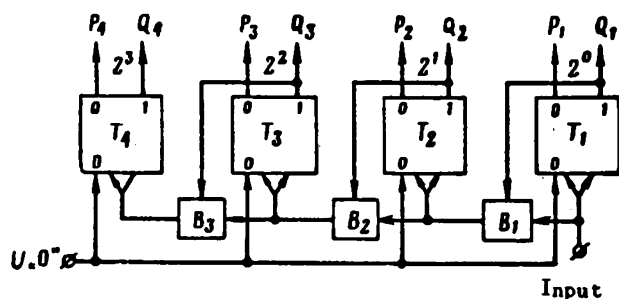


Fig.73 Continuous-Carry Counter

Consider the work of a continuous-carry counter (Fig.73) in the case where the binary number 0111 is written in it, i.e., the flip-flops  $T_1$ ,  $T_2$ ,  $T_3$  are in the position 1, and from their right outputs a potential is taken to pass an input pulse to the gates  $B_1$ ,  $B_2$ ,  $B_3$ . The first input pulse passes through the gates  $B_1$ ,  $B_2$ ,  $B_3$  which are opened on the upper input, and flips the flip-flop  $T_4$  into position 1. In addition, this pulse flows into the counter inputs of the flip-flops  $T_1$ ,  $T_2$ ,  $T_3$ , switching them from the state 0 to the state 1. The counter reading is now 1000, i.e., it has increased by one over the last reading. The second input pulse does not pass the loop of continuous carry since the gate  $B_1$  is blocked at the upper input. This pulse passes only into the counter input of the flip-flop  $T_1$ , switching it into position 1. The counter reading will now be 1001. The third input pulse, passing through the open gate  $B_1$ , flips the flip-flop  $T_2$  into position 1, and, in addition, the flip-flop  $T_1$  into position 0, and so on.

Thus, if the carry circuit is arranged in this manner, the input pulses appear almost simultaneously at the inputs of all flip-flops as far as the nearest flip-flop in position 0. This flip-flop will switch to position 1, and all the preceding flip-flops to position 0. This increases the counter readings by one. The continuous-carry circuit made up of gates has a very short delay time, and therefore its reset time after arrival of the regular input 149 pulse is only a little longer than the switching time of an individual flip-flop.

Reversible counters. Counters permitting counting in either the direction

of addition or the direction of subtraction are called reversible. A reversible counter is a combination of an add counter and a subtract counter.

Several versions of circuitry for reversible counters are used in digital computers. Let us consider the two most widely used counter types, based on static flip-flops.

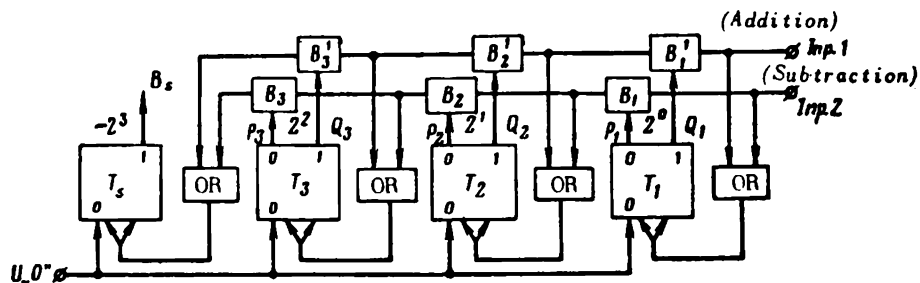


Fig.74 Reversible Static Flip-Flop Counter with Different Coupling for Addition and Subtraction of Pulses

Figure 74 shows the circuit of a three-digit static flip-flop reversible counter. Besides the flip-flops  $T_1$ ,  $T_2$ ,  $T_3$ , which take the values of the digit positions of the number, the counter also has a sign flip-flop  $T_4$  to take the sign of a number. The state zero of the sign flip-flop corresponds to a positive number, and the state one to a negative number in complement code. The counter operates in two regimes: in the regime of addition, when the pulses to be counted are fed to the first input of the counter, and in the regime of subtraction, when the pulses are fed to the second input. The pulse-potential gates  $B_i'$  ( $i = 1, 2, 3$ ) form a parallel carry line when the counter is operating in the addition regime. These gates are controlled by potentials taken from the right-hand outputs of the flip-flops. Similarly, the gates  $B_i$  ( $i = 1, 2, 3$ ), controlled by potentials from the left-hand outputs of the flip-flops, form a parallel-carry line when the counter is operating in the subtraction regime. Depending on the state of the gates  $B_i$  and  $B_i'$ , an input pulse may arise simultaneously at several flip-flops (if the delay due to the passage of the pulse through the gate is neglected). The pulses arrive at the counter input of the flip-flops through an OR circuit.

In the initial state, all flip-flops of a reversible counter are in zero position, the gates  $B_i'$  are closed at the lower input, while the gates  $B_i$  are open. If the counter is operating in the addition regime, then the first pulse fed to the input 1 does not pass into the parallel-carry line, but reaches the flip-flop  $T_1$  through the OR element and switches that flip-flop to position 1. As a result, the gate  $B_1'$  is opened at the lower input, while the gate  $B_1$  is closed. The second input pulse passes through the gate  $B_1'$  and switches the flip-flop  $T_2$  to position 1. It also sets the flip-flop  $T_1$  into position zero so that the gate  $B_1'$  is closed. The third pulse switches the flip-flop  $T_1$  to position 1 and reopens gate  $B_1'$ . The fourth input pulse passes through the open gates  $B_1'$  and  $B_2'$  and flips the flip-flop  $T_3$  into position 1. It



also switches the flip-flops  $T_1$  and  $T_2$  into position zero, and so on. Consequently, a reversible counter in the addition regime operates in the same way as a parallel-carry add counter.

In the subtraction regime, when the pulses arrive at the second input, the counter will operate similarly. Since there is a parallel-carry line, the input pulse arrives practically simultaneously at the inputs of all flip-flops

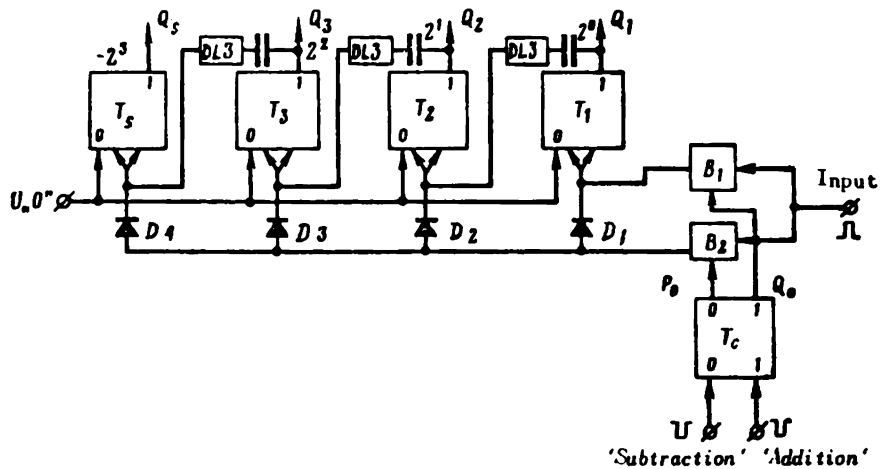


Fig.75 Reversible Static Flip-Flop Counter with Same Coupling for Addition and Subtraction of Pulses

up to the nearest flip-flop in position 1. This flip-flop is switched to the position 0 and the preceding flip-flops to the position 1. As a result, the counter reading is decreased by one. For example, if the binary number 100 is written in the counter, then the pulse arriving at the second input passes through the open gates  $B_1$ ,  $B_2$  and flops the flip-flops  $T_3$  into position 0. The pulse also arrives at the flip-flops  $T_1$  and  $T_2$  and switches them to the position 1. The number 011 will then be shown on the counter.

As already noted, negative numbers are fixed in the counter in complement code. Now let, in a counter operating in the subtraction regime, the binary number +010 (decimal number +2) be entered, corresponding to the states of the flip-flops given in the first row of Table 15. After feeding the first two input pulses, the counter is set in zero position. On arrival of the third pulse, all the flip-flops are set in position 1, and the counter will show the number 1 which, in the complement code, will be represented as  $[-1]_{c.o.} = 1.111$ . On arrival of the next pulses, the counter still operating in the subtraction regime will successively record the negative numbers -2, -3 etc. in the complement binary code.

In the reversible counter (Fig.75), in contrast to that discussed earlier, the flip-flops are connected in the same way for operation in the addition regime as in the subtraction regime. The input pulses pass either through the gate  $B_1$  on addition or through the gate  $B_2$  on subtraction, depending on the

TABLE 15

SEQUENCE OF OPERATION OF COUNTER ON FORMATION OF COMPLEMENT  
CODES OF NEGATIVE NUMBERS

Flip-Flop Weight of Digit	$T_0$	$T_1$	$T_2$	$T_3$
Decimal Numbers	$-2^0$	$2^0$	$2^1$	$2^2$
+2	0	0	1	0
+1	0	0	0	1
0	0	0	0	0
-1	1	1	1	1
-2	1	1	1	0
-3	1	1	0	1
.	.	.	.	.
.	.	.	.	.

position of the control flip-flop  $T_0$ . To switch the control flip-flop, addition and subtraction signals are fed to its inputs. Pulses are added in the same way as in the former system. In subtraction, the input pulses are fed through the blocking diodes  $D_1 - D_4$  simultaneously to the inputs of all flip-flops. This corresponds to the addition of the number 1.111, i.e., the complement code of minus one, which is equivalent to subtracting one. If, for example, the binary number 0.101 (the decimal number +5) is set up in the counter, the following result is obtained on adding it to the complement code of minus one:

$$\begin{array}{r} 0.101 \\ + 1.111 \\ \hline 0.100 \end{array}$$

i.e., the decimal number +4 is recorded. Owing to the replacement of the operation of subtraction by the operation of addition in complement code, only 152 one input is used in each flip-flop.

In the regime of subtraction, the operation of addition of a complement code minus one is performed in two half-periods. First, the input pulse simultaneously passing through the diodes  $D_1 - D_4$ , switches all the flip-flops after which the delayed carry pulses arrive at the inputs of the flip-flops of the higher-order digit positions, tripping them a second time. Thus, in the system of this counter, delay lines are in principle required to delay the carry pulses until the transient processes in the flip-flops, caused by the input pulse in the first half-period, have been completed. In the second half-period, carry pulses may arrive and be successively transmitted so that, in the worst case, the setup time of the counter will be  $n(\tau + t_{p1})$ , where  $n$  is the number of places in the counter,  $\tau$  the switching time of the flip-flop from one stable state to the other, and  $t_{p1}$  the pulse delay time in the delay line. In this case the relation  $t_{p1} \geq \tau$  must be maintained. The presence of a delay line decreases the speed of the counter in both subtraction and addition of pulses.

The counter readings are usually read out in the form of potential levels on the collector of the right (or left) triode of each flip-flop if the flip-flops are built of semiconductor triodes, i.e., are transistorized. If the potentials on the right-hand outputs of the flip-flops correspond to the direct code of the numbers recorded in the counter, then the potentials on the left-hand outputs of the flip-flops correspond to the inverse code.

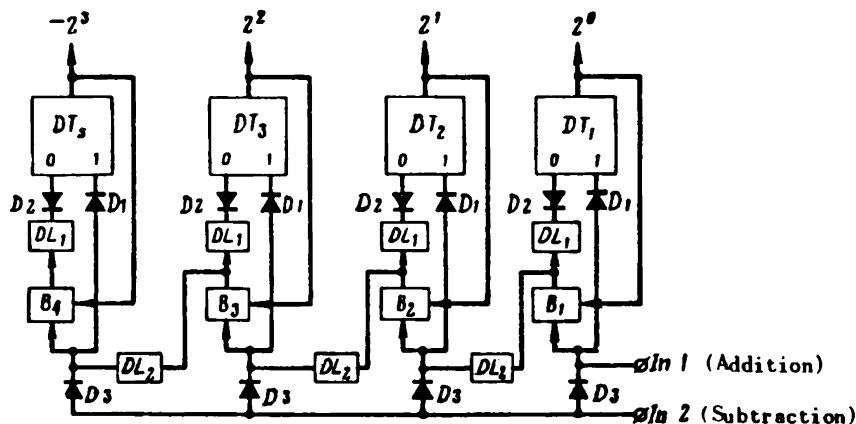


Fig.76 Reversible Counter using Dynamic Flip-Flops

Dynamic flip-flop counter. In a reversible counter with dynamic flip-flops (Fig.76), each flip-flop is based on the circuit shown in Fig.67. To form the counting input of the flip-flop, a gate is used. The counter can 153 operate in the regime of addition and in that of subtraction, and the principle of operation in the regime of subtraction is the same as that of the static flip-flop counter (Fig.75).

The pulses fed to the input 1 in addition, or to the input 2 in subtraction, are of positive polarity. If the flip-flop of the counter is in position 0, then the input pulse can pass only through the diode  $D_1$  to the one input of the flip-flop, since the gate  $B_i$  ( $i = 1, 2, 3, 4$ ) is closed. The flip-flop will be switched to position 1, and the gate will be open to pass the next input pulse. The following input pulse passes first to the one input, without changing the position of the flip-flop. It also passes through the open gate and arrives at the zero input of the flip-flop, switching it into the zero position. The delay line  $DL_1$  is designed to separate in time the pulses arriving at the one and zero inputs of the flip-flop.

Let the counter be in the initial state, i.e., let all flip-flops be in the position 0. In the addition regime, the first input pulse flips the flip-flop  $DT_1$  into position 1 and opens the gate  $B_1$ . The second input pulse passes through the gate  $B_1$  and sets the flip-flop  $DT_1$  into position 0. This same pulse also is fed through the delay line  $DL_2$  and the diode  $D_1$  to the one input of the flip-flop  $DT_2$ , switching it into position 1. The counter will then read two. The gate must have a transformer output, in order to receive pulses of either positive polarity (carry pulses to the next higher-order digit position)

or negative polarity (for transfer to the zero input of the flip-flop). On arrival of the third input pulse, the flip-flop  $DT_1$  is flipped into position 1, while the remaining flip-flops keep their state; i.e., the number three is now recorded in the counter, etc.

In the subtraction regime, the input pulse passes simultaneously through the diodes  $D_3$  to all flip-flops, which is equivalent to adding the complement code of minus one or to subtracting one.

The delay line  $DL_2$  delays the carry pulses to the highest digit position for the time of the transient processes in the flip-flops, that are due to the input pulses when the counter is operating in the subtraction regime.

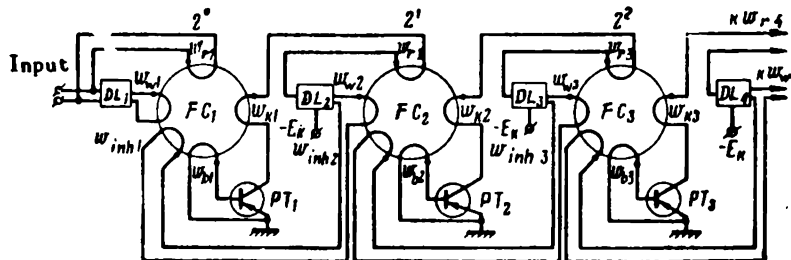


Fig.77 Binary Counter using Ferrite-Transistor Cells

Counter using ferrite-transistor cells. In a counter with ferrite-transistor cells (Fig.77), each cell serves to represent one digit of the number to be set up, and the cells of lower digit positions are located to the left.

The collector winding  $w_k$  of each core is connected to the read winding  $w_r$  of the next core and also, through the delay line, in series with its write winding  $w_w$  and with the inhibit winding  $w_{inh}$  on its own core.

Let all the cells of the counter be in the zero position. When the first pulse arrives at the counter input, it passes first into the read winding  $w_r$  of the core  $FC_1$ , without changing its magnetic state. After a certain time, determined by the pulse latency time of the delay line  $DL_1$ , the pulse appears in the write winding  $w_{r1}$  which switches the core  $FC_1$  into position 1. /15/

The second input pulse again arrives first at the winding  $w_{r1}$  and this time does switch the core  $FC_1$  from position 1 to position 0. The triode  $PT_1$  starts conducting, and a carry pulse arises in the collector winding  $w_{k1}$ . This pulse first arrives at the winding  $w_{r2}$  of the core  $FC_2$  without changing its state and then, after passing the delay line  $DL_2$ , at the write winding  $w_{w2}$  of that core setting it in position 1, after which the pulse arrives at the inhibit winding  $w_{inh1}$  of the first core. By this time, the second input pulse of the counter, after passing delay line  $DL_1$ , enters the write winding  $w_{w1}$ . The action of this pulse is compensated by the action of the current pulse in the inhibit winding. Consequently, after the second pulse is fed to the counter input, the

core  $FC_2$  will be in the state 1, and the cores  $FC_1$  and  $FC_3$  in the state 0, i.e., the binary number 010 will be recorded on the counter.

By similar reasoning, it can be proved that, after the third input pulse, the cores  $FC_1$  and  $FC_2$  will be in position 1 and the core  $FC_3$  in position 0. After the fourth input pulse, the cores  $FC_1$  and  $FC_2$  are switched into position 0, and the core  $FC_3$  into position 1, and so on. It must be borne in mind that when a pulse arrives from the collector winding of the preceding core into the read and write windings of the following core, belonging to the cell of the next higher digit position, then the stable state of magnetization of that cell can change only once. This is due to the fact that, after a certain time following the switching of a core from position 1 to position 0, a current pulse compensating the action of the pulse in the write winding will always appear in its inhibit winding. /155

Thus, the positions occupied by the cores of ferrite-transistor cells correspond to the binary code of the number of pulses arriving at the counter input. Like the flip-flop cell, each core of the counter is returned to its initial state with every second pulse arriving at its write winding. The number of series-connected ferrite-transistor cells and, consequently, the number of digit positions of a counter can be rather great, since the triodes amplify the signals arriving at the cells of the higher digit positions.

## Section 26. Registers

A device serving to remember one binary number is known as a register. If a register has additional elements, the number stored in it can be read out either in parallel or in serial code, meaning that registers are also used to convert a parallel code into a serial code, and vice versa. If necessary, in multiplication, division, and other operations, the number in the register can be shifted one or several places to the right or left. To read out a number from a register and to shift numbers, readout pulses are used; they are also called shift pulses, and the registers themselves are known as shift registers.

Consider the common circuit of the register (Fig.78) used for storage of a three-place binary number. The number is entered in the register and read /156 out from it in the form of parallel code. Such a register is called a parallel-action or parallel register. In describing register systems using static flip-flops, a flip-flop as shown in Fig.62 will be meant, provided that a high potential on the collector of the left triode corresponds to the state 1 of the flip-flop and a high potential on the collector of the right triode to the state 0.

The code of the number to be recorded in the register is represented in the form of the potential-level code transfer buses. A high potential corresponds to code "1" and a low potential to code "0".

When the write control pulse CP-1 is fed to the input gates  $B_1$ , pulses representing the code of the number will arrive at the right-hand inputs of the register flip-flops. These latter are then switched to the corresponding stable states, which they may retain as long as desired.

The number stored in the register is transferred out or read out through the output gates  $B_2$ , as soon as the read pulse CP-2 is fed to them. The read-out of the number can be repeated as often as desired, since the state of the register flip-flops is maintained. A number is erased by feeding a clearing pulse to the left-hand inputs of the flip-flops; on arrival of this pulse all flip-flops are set in the code position "0".

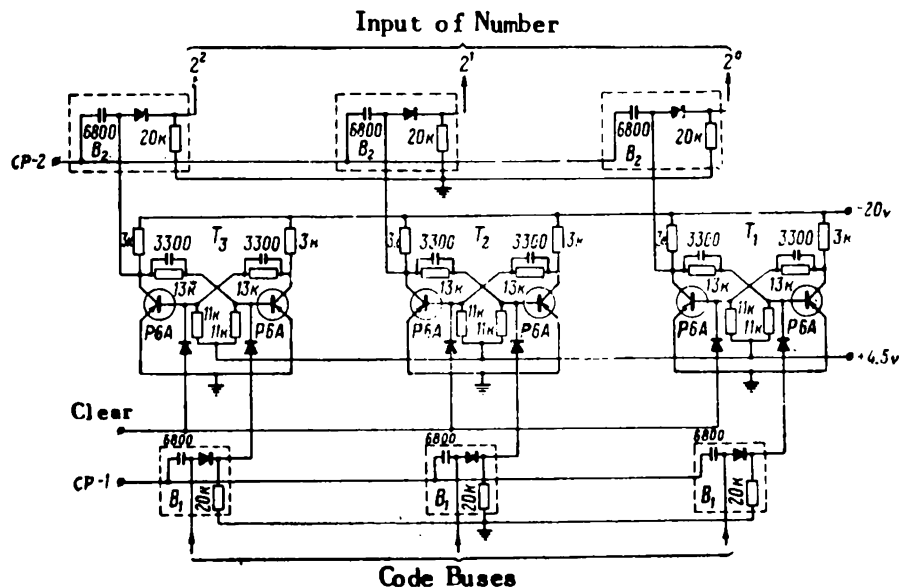


Fig. 78 Parallel-Action Register using Static Flip-Flops

Parallel registers are used in arithmetic, memory, and other units of digital computers. The advantages of such registers when used as storage devices for a single number are as follows:

- high speed (for writing, erasing, or readout of a number, not more than  $2 \mu\text{sec}$  are required);
- repeated readout of a number without its distortion in the register;
- possibility of representing a number read out from the register either in direct code or in inverse code if both flip-flop outputs are used.

Shift registers are widely used, especially in arithmetic units of digital computers. Figure 79 shows a three-digit shift register with static flip-flops. Here, the flip-flops are connected in series, the collector of the right-hand flip-flop of the preceding cell being coupled to the counting input of the following cell through a delay line.

Let the binary number 111 be stored in the register. To shift it one place to the right, one pulse must be fed to the shift bus. Since this pulse is an erase signal for all the cells, all will simultaneously be reset to the zero state. In this case, a positive voltage pulse will arrive at the outputs of the flip-flops and will pass through a delay line to the counting input of

the following flip-flop, switching it to the state 1. Thus, after the first pulse passes on the shift bus, the register will show the code of the number 011 instead of the code of the number 111, and a signal corresponding to the code "1" will appear at the output. After passage of the second shift pulse, the register will now contain the code of the number 001, and a second signal, corresponding to the code "1", will appear at its output. Finally, after passage of the third shift pulse, the position of the register flip-flops will correspond to the code of the number 000, and the serial code of the number 111 stored in the register will be obtained at its output.

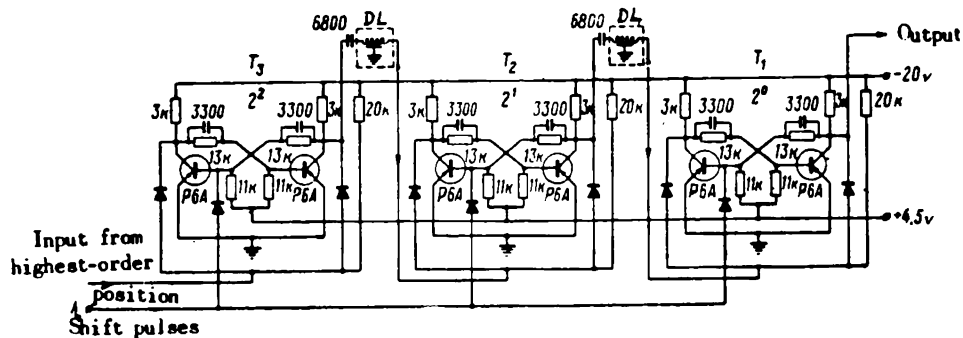


Fig. 79 Static-Flip-Flop Shift Register

If the code "0" were stored in the preceding cell of the register, then during shift of the number, after passage of the shift pulse, the pulse will not be transmitted into the delay line, and the code of "0" will be set up in the next cell. On transition of a cell from the state 0 to the state 1 a negative voltage pulse arises at its output and, after passing the delay line, will arrive at the counting input of the next cell. The state of that cell, however, is not changed thereby, since it reacts only to positive pulses.

It is obvious from our analysis of the operation of a shift register that each cell must be so laid out that the frequency of its operation (transition from one stable state to the other) shall be twice the repetition rate of the shift pulses. During a single interval between the shift pulses, the cell will operate once, on clearing, and then again on transfer of the pulse from the delay line to its input.

The delay lines in the shift register perform the function of storage elements for the pulse arriving from the output of the preceding cell at the input of the following cell. The pulse delay time is so selected that, at the time of arrival of a pulse from the delay line to the flip-flop input, all transient processes in it shall have ceased. This time usually does not exceed a value of 0.3 - 0.5  $\mu$ sec. In vacuum-tube shift registers, the frequency of the shift pulses may go as high as 1 Mc.

Figure 80 gives the circuit diagram of a register for converting the space-positional (parallel) code of a number into a time-pulse (serial) code, and vice versa. The register consists of several flip-flops connected in

series.

Let us first consider the operation of a register in converting the parallel code of a number into its serial code. As in the register shown in Fig.78, the number, represented in parallel code, is recorded when the write pulse CP-1 is fed to the input gates  $B_1$ . In this case, the code of a number arriving on the code bus is passed through the gates and entered in the cells of the register. For readout of the number (in this case, a three-digit number) in serial

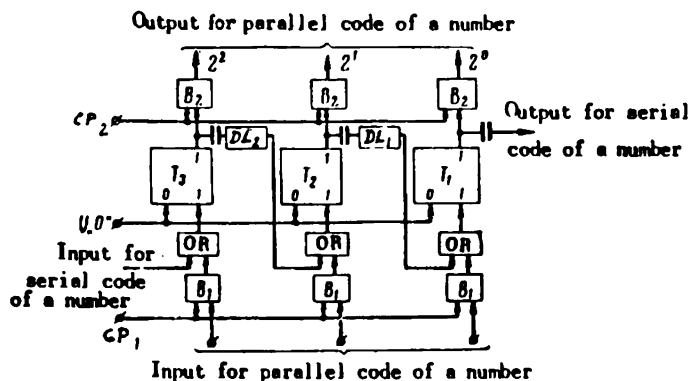


Fig.80 Register for Converting the Parallel Code of a Number into the Serial Code, and Vice Versa

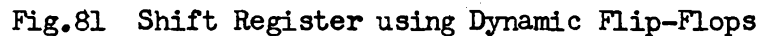
code, shift pulses are fed at definite time intervals to the shift bus. The register here operates like the register of Fig.79.

The serial code of a number is converted into the parallel code in the following way: During writing, the code is fed, with the lowest digit positions first, through the OR circuit into the counting input of the extreme left cell  $T_3$ . After the first (least significant) digit of the number has been fed to the cell  $T_3$ , the first shift pulse is introduced; this pulse copies the code of the lowest digit of the number from cell  $T_3$  into cell  $T_2$ . Then, in cell  $T_3$ , the code of the second digit of the number is written, after which the second shift pulse is fed, which copies the codes of the first and second places from the cells  $T_2$  and  $T_3$  into the cells  $T_1$  and  $T_2$ , respectively. Finally, the code of the third digit of the number is written into the cell  $T_3$ , and thus the entire number has been recorded in the register. To put out this number in the parallel code, i.e., all digits simultaneously (each digit having its own code bus), a readout pulse CP-2 must be fed to the output gates  $B_2$ . The readout 159 can be repeated many times, since the information stored in the register is not distorted thereby. A record is erased by shifting pulses (in this case three such pulses must be given) or by applying a wide clearing pulse.

Let us consider the sequence of operation of a shift register built of dynamic flip-flops (Fig.81). Let the number 000 be stored in the register in the initial state, i.e., let no sync pulses pass to the output of the flip-flops. If the code signal 1 is fed to the ones input of the flip-flop  $DT_3$ , that flip-flop will be switched to the state 1, and at its output there will



The first shift pulse of negative polarity passes through the inverters  $HE_1$  and  $HE_2$  and then arrives at the gates  $B_1$  and  $B_2$  in the form of a pulse of positive polarity. Since the flip-flop  $DT_2$  is in the position 0, and the flip-flop  $DT_3$  in the position 1, the shift pulse passes through the gate  $B_2$  and arrives in the delay line  $DL_2$ . In addition, the first shift pulse is fed



The operation of a dynamic flip-flop register obeys the following law: On arrival of the regular shift pulse, each successive flip-flop is set in the 160 position that the preceding flip-flop had before arrival of that pulse, i.e., each shift pulse, as it were, transmits the code of the higher-order position to the flip-flop of the next lower-order position and thus shifts the entire code of the number one place to the right.

Shift registers with ferrite-transistor and ferrite-diode cells are widely used, especially in the circuitry of special-purpose digital computers. The circuit given in Fig.37a, having three ferrite-transistor cells with single-cycle feed connected in series, is essentially a circuit of a three-digit shift register. The input of the register is the input of the first cell. The read-

out pulses (clock pulses) play the role of shift pulses. When a binary number is written into the register, it is fed in serial code to the write winding of the core in the first cell. After each code pulse, a readout pulse is introduced, shifting the number in the registers one place to the right.

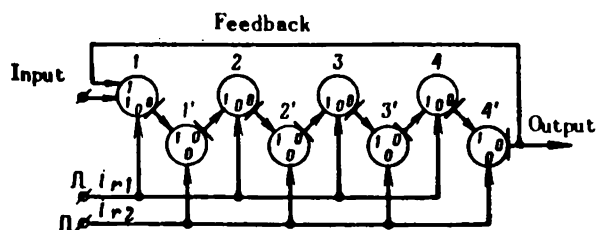


Fig.82 Shift Register using Ferrite-Transistor Cells

The advantage of shift registers using ferrite-transistor cells with single-cycle feed is that they are rather simple and require only one source of clock pulses. The disadvantage is their low operating speed and the losses of energy on the delay line.

Shift registers using ferrite-transistor cells in push-pull feed are widely used. Figure 37b shows a circuit for series connection of three ferrite-transistor cells in push-pull, and Fig.82 is the circuit of a shift register in push-pull feed with four digit positions, with symbolic representation of the cells. The numerals 1 and 0 inside the circles indicate that, on passage of pulses through the corresponding loops indicated by straight-line arrows, the cell is set in either position 1 or position 0. /161

To represent the code of each digit of a binary number in a push-pull shift register, two ferrite-transistor cells are necessary: one, known as the main cell for storage of the code of one digit of the number, and the other, known as the auxiliary cell for temporary storage of the code of the digit during the shifting of the number. Figure 82 shows the main cells of the registers, denoted by the numerals 1, 2, 3, 4 and the auxiliary cells by the numerals 1', 2', 3', 4'.

Let the binary number 111 be recorded in cells 1, 2, and 3 of the register. To shift the number one place to the right, the readout pulse  $i_{r1}$  must first be fed, on whose passage the cells 1, 2, and 3 are set into the zero position and the cells 1', 2', 3' into the one position, i.e., the number 111 is transferred from the main cells to the auxiliary cells. Then, the readout pulse  $i_{r2}$  will switch the cells 1', 2', and 3' to the zero position and the cells 2, 3, and 4 into the one position. As a result, the number has now been shifted one place to the right. Thus, the advance of the number code along the register is accomplished by alternate feeding of the readout pulses  $i_{r1}$  and  $i_{r2}$ , shifted a half-period with respect to each other.

The practical circuits of push-pull shift registers with ferrite-transistor cells operate at clock-pulse repetition rates up to 100 - 150 kc, but

require a greater number of cells than one-cycle circuits.

Shift registers with ferrite-diode cells are constructed in exactly the same way as those with ferrite-transistor cells. The circuits in Figs. 25 and 26, showing series connection of ferrite-diode cells in push-pull with single-cycle feed, give a general idea of how a shift register using such cells is built. For example, the circuit in Fig. 25 is a single-cycle circuit of a three-digit shift register. Because of the drawbacks of ferrite-diode cells, such registers are seldom used.

All shift registers, regardless of the constituent elements, can be so designed as to store the recorded information. In the general case, on arrival of the shift pulses, the number recorded in the register is read out in serial code and all the cells are set into the code position "0", i.e., the information is erased. If, however, the output of the register is connected with its input, i.e., if a feedback is established, then the passage of shift pulses will cause the code of each digit to arrive not only at the output of the register but also at the input for rewriting (see Fig. 82). Thanks to this, at continuous arrival of clock or shift pulses, the recorded number will circulate in the register, i.e., the information will not be erased.

## Section 27. Selective Circuits (Decoders)

/162

A selective circuit, or decoder, is an  $(n, k)$ -pole which, for a given combination of binary signals at its inputs, forms the code signal 1 only at a single completely determinate output. An  $(n, k)$ -pole means a circuit with  $n$  inputs and  $k$  outputs. The relation between  $k$  and  $n$  in decoders is of the form  $k = 2^n$ .

The binary signals fed to the decoder inputs are combinations of zeros and ones, i.e., binary numbers. If the outputs of a decoder are designated by the letters  $P_j$  ( $j = 0, 1, 2, \dots, k - 1$ ), then the code signal 1 appears at the output, for which the value of the subscript  $j$  in the output denotation is the same as the binary number corresponding to the input combination of zeros and ones. For example, if signals representing the binary number 1101 (the decimal number 13) are fed to a decoder with four inputs, then the code signal 1 will appear at the output  $P_{13}$ . Essentially, a decoder affects the decoding of a code fed to its input by delivering a control signal to one of its outputs.

The operation of a decoder with  $n$  inputs and  $k$  outputs is described by the equations:

$$\begin{aligned} P_0 &= \bar{x}_{n-1} \bar{x}_{n-2} \dots \bar{x}_1 \dots \bar{x}_2 \bar{x}_1 \bar{x}_0; \\ P_1 &= \bar{x}_{n-1} \bar{x}_{n-2} \dots \bar{x}_1 \dots \bar{x}_2 \bar{x}_1 x_0; \\ P_2 &= \bar{x}_{n-1} \bar{x}_{n-2} \dots \bar{x}_1 \dots \bar{x}_2 x_1 \bar{x}_0; \\ P_3 &= \bar{x}_{n-1} \bar{x}_{n-2} \dots \bar{x}_1 \dots \bar{x}_2 x_1 x_0; \end{aligned} \tag{47}$$

(cont'd.)

$$P_{k-1} = x_{n-1}x_{n-2} \dots x_l \dots x_2x_1x_0,$$

where  $x_i$  ( $i = 0, 1, 2, \dots, n - 1$ ) are the inputs of the decoder;  
 $P_j$  ( $j = 0, 1, 2, \dots, k - 1$ ) are the outputs of the decoder.

Decoders are used in a digital computer to decode codes and feed signals into arbitrary control loops. For example, they are used in the control units of digital computers to decode the operation code and send a control signal into

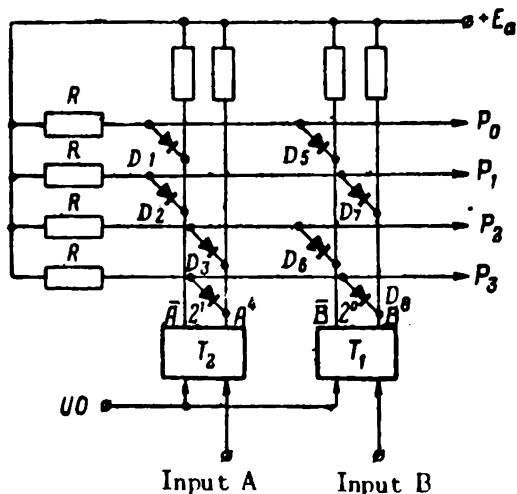


Fig.83 Rectangular Diode Decoder with Two Inputs

all loops connected with the elements and units of the computer that take part in performing the given operation. The cells of the storage units of the computer are selected by the aid of decoders, for writing, reading, and regenerating the codes of numbers and instructions.

To rate decoders of various types, the following characteristics are used:

- a) Speed, determined by the time required to decode the code of one number.
- b) Economy, determined by the method of constructing the decoder and the type of its constituent elements.
- c) Reliability of operation, which depends both on the type of elements used in it, and on their number. The reliability of operation of a decoder /163 is sometimes evaluated from the ratio of the code signal 1 appearing at one of its outputs to the maximum noise appearing at the other outputs. The larger this ratio, the higher will be the quality of the decoder.

Various elements are used in decoders: semiconductor diodes, ferrite cores with a rectangular hysteresis loop, ferrite-transistor cells, diodes in conjunction with pulse transformers, etc. Decoders using semiconductor diodes

(diode decoders) and decoders using ferrite cores or ferrite-transistor cells (magnetic decoders) are the most popular. The codes of the numbers fed to the input of a decoder are usually read out of registers.

Diode decoders. To illustrate the operating principle of diode decoders, let us first consider an elementary circuit with two inputs (Fig.83). The circuit is based on a network (matrix) formed by horizontal and vertical wires (buses) and semiconductor diodes connecting them at certain points. The circuit has two inputs (A, B) and four outputs ( $P_0, P_1, P_2, P_3$ ) and operates according to a truth table (Table 16).

TABLE 16  
LOGIC OF OPERATION OF A TWO-INPUT DIODE DECODER

Inputs		Outputs			
A	B	$P_0$	$P_1$	$P_2$	$P_3$
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

If, for example,  $A = 0$  and  $B = 0$ , then both flip-flops of the register 164 are in the position 0. In this case, there are high potentials on the left-hand outputs of the flip-flops so that the diodes  $D_1, D_2$  as well as  $D_5$  and  $D_6$  are cut off, while the other diodes are conducting. If at least one of the diodes connected to any output bus is open, then the potential of that bus will be low since practically all the voltage from the source  $+E_a$  falls across the resistance  $R$ , which is considerably greater in value than the resistance of the diode in the forward direction. Consequently, a high potential corresponding to the code "1", appears only on the output bus  $P_0$  since the diodes  $D_1$  and  $D_5$  connected with that bus are in the nonconducting state.

If  $A = 1$  and  $B = 1$ , then the diodes  $D_3, D_4, D_7$  and  $D_8$  will be cut off. Consequently, the high potential will be transferred only to the bus  $P_3$ , since the diode  $D_4$  and  $D_8$  connected with it are cut off.

It is easy to demonstrate that, at the combination  $A = 0, B = 1$ , the high potential is transferred to the output  $P_1$  and, for the combination  $A = 1, B = 0$ , to the output  $P_2$ .

The logic functions describing the operation of the decoder have the following form:

$$P_0 = \bar{A}\bar{B};$$

$$P_1 = \bar{A}B;$$

$$P_2 = A\bar{B};$$

$$P_3 = AB.$$

A two-input diode decoder can be represented by four independent coincidence circuits with two inputs each, under the condition that the values of the input signals are inverted. In Fig.83, the two-input AND elements are formed by diodes coupled to the horizontal (output) buses.

Diode decoders with a larger number of inputs can also be constructed by this principle. For example, Fig.84 gives the circuit of a four-input decoder. Considering that, to the code "0", there corresponds a position of the flip-flop such that a high potential is present at its left-hand output and the code "1" at its right-hand output, it is obvious that the output bus whose number is the same as the number recorded in the register will be under a high potential. In this case, the lowest-order position of the number will be stored

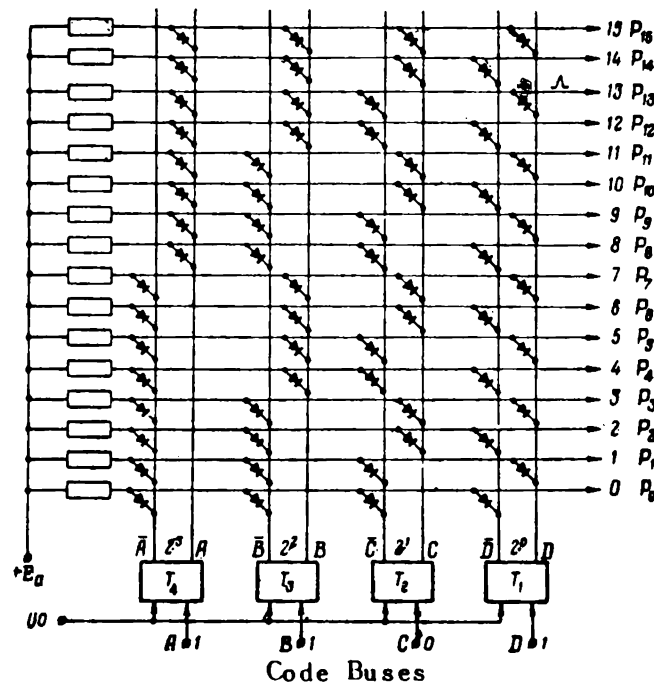


Fig.84 Rectangular Diode Decoder with Four Inputs

in the flip-flop  $T_1$ , and the highest-order position in the flip-flop  $T_4$ . If the number 1101 is stored in the register (the decimal number 13), then all the diodes connected to the output bus  $P_{13}$  will be cut off. As for all the other output buses, each of them is connected to at least one diode in the conducting state for the given state of the register flip-flops. Consequently, the bus  $P_{13}$  is excited. In such a decoder, the diode network is composed of sixteen independent coincidence circuits with four inputs each. One coincidence circuit /16 is composed of the diodes connected with the same output bus.

Decoders built on this principle are called rectangular or single-stage. In the general case, the matrix of a rectangular decoder consists of  $2n$  vertical buses, to which noninverted and inverted code signals (or the paraphase code of

the number) are fed, and of  $2^n$  output buses, where  $n$  is the number of digits in the binary number. To build such a matrix we need  $N_1 = n \times 2^n$  diodes. This makes it obvious that, with increasing  $n$ , the number of diodes in the matrix increases sharply. It is therefore expedient to build rectangular diode decoders for a small number of inputs, not greater than four.

Two-stage and multistage diode decoders are more economical, especially for large values of  $n$ .

Let us consider the design principle of a two-stage diode decoder with four inputs  $A$ ,  $B$ ,  $C$ , and  $D$ . The operation of such a decoder is described by /166 the equations:

$$\begin{aligned}
 P_0 &= \overline{A}\overline{B}\overline{C}\overline{D}; & P_8 &= A\overline{B}\overline{C}\overline{D}; \\
 P_1 &= \overline{A}\overline{B}\overline{C}D; & P_9 &= A\overline{B}\overline{C}D; \\
 P_2 &= \overline{A}\overline{B}C\overline{D}; & P_{10} &= A\overline{B}C\overline{D}; \\
 P_3 &= \overline{A}\overline{B}CD; & P_{11} &= A\overline{B}CD; \\
 P_4 &= \overline{A}B\overline{C}\overline{D}; & P_{12} &= AB\overline{C}\overline{D}; \\
 P_5 &= \overline{A}B\overline{C}D; & P_{13} &= AB\overline{C}D; \\
 P_6 &= \overline{A}BC\overline{D}; & P_{14} &= ABC\overline{D}; \\
 P_7 &= \overline{A}BCD; & P_{15} &= ABCD.
 \end{aligned} \tag{48}$$

We now introduce the notation:

$$\begin{aligned}
 A_1 &= \overline{A}\overline{B}; & B_1 &= \overline{C}\overline{D}; \\
 A_2 &= \overline{A}B; & B_2 &= \overline{C}D; \\
 A_3 &= A\overline{B}; & B_3 &= C\overline{D}; \\
 A_4 &= AB; & B_4 &= CD.
 \end{aligned}$$

Then, eqs.(48) take the form:

$$\begin{aligned}
 P_0 &= A_1B_1; & P_8 &= A_3B_1; \\
 P_1 &= A_1B_2; & P_9 &= A_3B_2; \\
 P_2 &= A_1B_3; & P_{10} &= A_3B_3; \\
 P_3 &= A_1B_4; & P_{11} &= A_3B_4; \\
 P_4 &= A_2B_1; & P_{12} &= A_4B_1; \\
 P_5 &= A_2B_2; & P_{13} &= A_4B_2; \\
 P_6 &= A_2B_3; & P_{14} &= A_4B_3; \\
 P_7 &= A_2B_4; & P_{15} &= A_4B_4.
 \end{aligned} \tag{49}$$

The logic functions  $A_1 - A_4$  are formed by combinations of the arguments  $A$ ,  $B$ , and their negations, while the functions  $B_1 - B_4$  are formed by combinations of the arguments  $C$ ,  $D$ , and their negations. In turn, the logic functions  $P_0 - P_{15}$  are formed by combinations of the functions  $A_1 - A_4$  and  $B_1 - B_4$ . The first stage of a two-stage decoder consists of coincidence circuits realizing the functions  $A_1 - A_4$  and  $B_1 - B_4$  and the second stage, of coincidence circuits

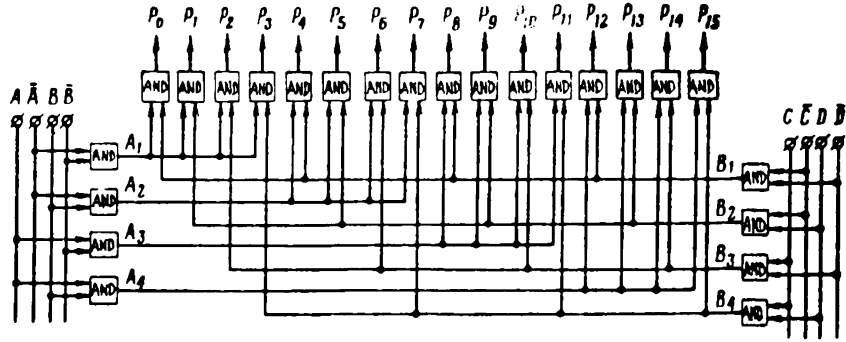


Fig.85 Function Circuit of a Two-Stage Decoder with Four Inputs

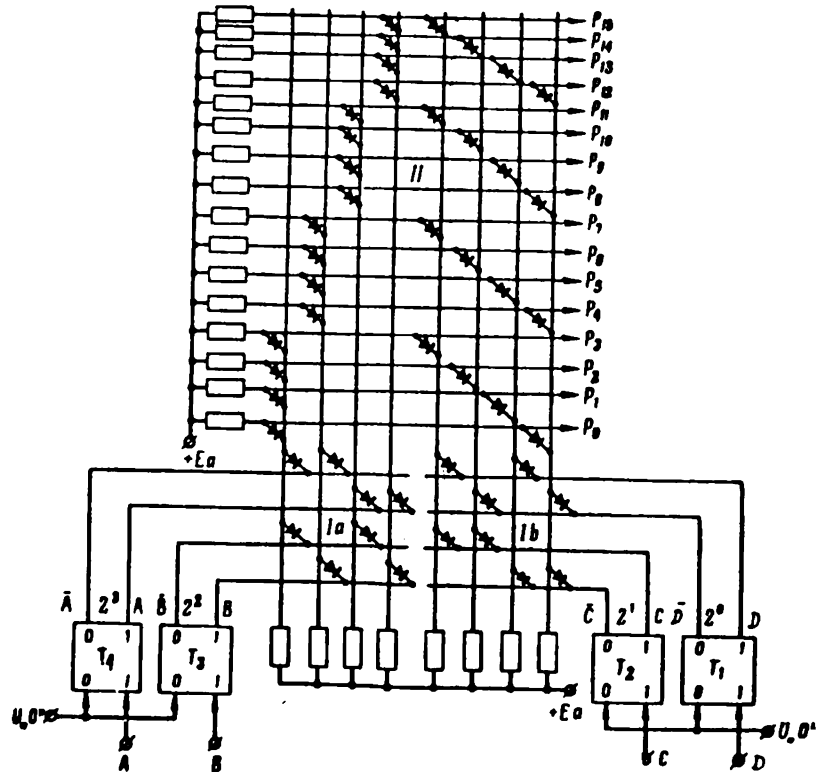


Fig.86 Two-Stage Diode Decoder with Four Inputs



realizing the functions  $P_0 - P_{15}$  [or eqs.(49)]. Figures 85 and 86, which are respectively a function circuit and a schematic diagram of a two-stage decoder with four inputs, clearly show these stages. In the circuit diagram, the first stage is composed of the diode matrices 1a and 1b with four outputs each and the second stage, of the matrix II with the 16 outputs  $P_0 - P_{15}$ . Depending on the code of the number recorded in the register, one of the output buses of the matrices 1a and 1b is excited. In this case, only one of the output buses of the matrix of the second stage is connected to the two diodes in the nonconducting state. On this bus there will be a high potential and a low potential 168 on the others.

A two-stage decoder with  $n$  inputs is built as follows: The number  $n$  is divided into two groups such that  $\frac{n}{2}$  shall be in each group if  $n$  is even, or  $\frac{n+1}{2}$  and  $\frac{n-1}{2}$  if  $n$  is odd. Then, by the aid of coincidence circuits, the functions representing various combinations of the input variables and their negations are formed in each group. The second stage is also composed of AND elements, the input signals for these being the output signals of the coincidence circuits of the first stage.

The total number of diodes in a two-stage diode decoder of  $n$  inputs is determined by the formulas:

a) For even  $n$ :

$$N_2 = n \cdot 2^{\frac{n}{2}} + 2 \cdot 2^n;$$

b) For odd  $n$ :

$$N_2 = \frac{n+1}{2} \cdot 2^{\frac{n+1}{2}} + \frac{n-1}{2} \cdot 2^{\frac{n-1}{2}} + 2 \cdot 2^n.$$

Multistage decoders are the most economical of all types of diode decoders.

A multistage decoder consists of several series-connected two-stage decoders. For this construction, at first the  $n$  input variables are divided into two groups as was done in the construction of the two-stage decoder. Then each group is divided into subgroups in the same way until all the subgroups contain either two or three variables.

For subgroups with two or three variables, diode matrices with four and eight outputs respectively are formed, as shown in Fig.83 for two input variables  $A$  and  $B$ . These matrices compose the first stage of the decoder. Then the outputs of each pair of diode matrices of the first stage are combined by the aid of diodes, resulting in the formation of diode matrices of the second stage. Next, the outputs of each pair of second stage matrices are combined, thus forming third-stage matrices. This process is continued until the last

stage of the decoder, with only a single diode matrix, is formed.

For the example shown in Fig.87, we give the symbolic representation of a circuit for a diode decoder of multistage type with 1024 outputs. Each diode matrix is shown in the form of a square; the numeral inside the square indicates the number of output buses of the matrix. It should be noted that only one output bus of each matrix is shown in the diagram.

The register has 10 flip-flops, since there are 10 digits in the binary 169 numbers. These are divided into two subgroups of five flip-flops each. In turn, each subgroup is divided into two parts, with two flip-flops in one part and three in the other. The diode matrices in the first stage have either four or eight output buses; these matrices are controlled by two or three flip-flops respectively.

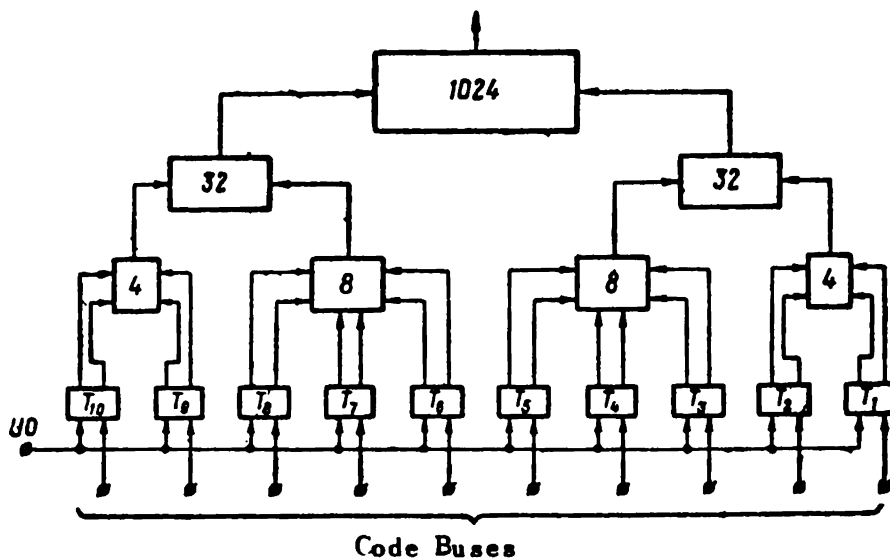


Fig.87 Multistage Diode Decoder with 1024 Outputs

The outputs of the matrices with four or eight outputs are connected by means of diodes forming a matrix of 32 outputs. Two such matrices compose the second stage of the decoder. Finally, a matrix of 1024 outputs (the third stage) is formed by combining the output buses of two matrices, each with 32 outputs.

In a multistage diode decoder, any matrix is connected with each output bus by not more than three diodes, if the above order of arrangement is observed.

Diode decoders use a paraphase code to represent the binary numbers recorded in the input register of the decoder, i.e., both outputs of the flip-flops are used: The direct code of the number is read out from the right-hand outputs, and the inverse code from the left-hand outputs.

The above diode decoders are equal in operating speed: A control signal

will appear at the output with practically no delay. The switching time of the decoder is determined by the time required to erase the number previously recorded in the register and to write the new number. The economy of diode decoders of various types can be evaluated by analyzing the data of Table 17. /170

TABLE 17

NUMBER OF DIODES REQUIRED FOR THE CONSTRUCTION OF DECODERS

Number of Inputs $n$	Number of Outputs $k$	Rectangular Decoder		Multistage Decoder	
		Number of Stages	Number of Diodes	Number of Stages	Number of Diodes
2	4	1	8	1	8
3	8	1	24	1	24
4	16	1	64	2	48
5	32	1	160	2	96
6	64	1	384	2	176
7	128	1	896	3	328
8	256	1	2048	3	608
9	512	1	4608	3	1168
10	1024	1	10240	3	2240

It will be clear from Table 17 that, with increasing number of input variables  $n$ , the economy of multistage decoders increases substantially by comparison with that of rectangular decoders.

In several Soviet digital computers, so-called non-coincidence decoders (Fig.88) are used. Such a decoder realizes the logic operation of nonequivalence.

The device consists of a four-stage diode circuit whose inputs are connected to the outputs of two flip-flops: the inputs I-0 and I-1 to the outputs of the first flip-flop, and the inputs II-0 and II-1 to the outputs of the second flip-flop. The diodes  $D_1$ ,  $D_2$  and  $D_3$ ,  $D_4$  which enter into the first stage of the system form two decoders. At the output of the decoders, the diodes  $D_5$  and  $D_6$  are connected and form the second stage of the system, serving to realize the logic operation OR.

The output of the first decoder (diodes  $D_1$  and  $D_2$ ) will be at high potential if the flip-flop  $T_1$  connected to the input I-0 and I-1 is in position 1, and the flip-flop  $T_2$  connected to the inputs II-0 and II-1 is in position 0. A high potential in this case is established at the inputs I-1 and II-0, i.e., the diodes  $D_1$  and  $D_2$  are cut off.

The output of the second decoder (diodes  $D_3$  and  $D_4$ ) will be at high potential when the flip-flop  $T_1$  is in position 0 and the flip-flop  $T_2$  in position 1. In this case, the diodes  $D_3$  and  $D_4$  will be cut off, i.e., both diodes of the second decoder will be cut off. If both flip-flops are either in position 1 or in position 0, then the potential at the outputs of the decoders will

be low, since one of the diodes of each decoder will be in the conducting state. Consequently, at the output of the OR circuit (diodes  $D_5$  and  $D_6$ ) the potential will be high if and only if the flip-flops are in opposite states of equilibrium. /171

The diode  $D_7$  is the third stage of the system. Its cathode is fed with a control pulse of positive polarity, permitting operation of the non-coincidence decoder. If the control pulse is not fed and, consequently, the diode  $D_7$  is in the conducting state, then there will be a low potential at the circuit output, regardless of the state of the flip-flops; the non-coincidence decoder will be cut off. When the control pulse is fed, the diode  $D_7$  is cut off and the potential level at the circuit output is determined by the state of the flip-flops; the non-coincidence decoder is now conducting.

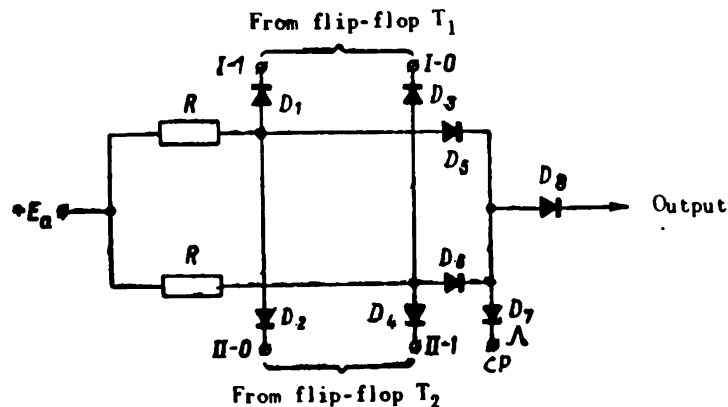


Fig.88 Non-Coincidence Decoder

The fourth stage of this system is the diode  $D_8$  which, together with the corresponding diodes of the other non-coincidence decoders, forms an OR circuit. This latter can be connected to the output of another logic circuit, for instance AND.

Thus the non-coincidence decoder is a logic circuit at whose output a high potential level can be obtained if two flip-flops connected to its inputs are in different states and if simultaneously, at an auxiliary input, a positive pulse is fed permitting the decoder to operate.

Magnetic decoders. Like diode decoders, magnetic decoders can be built as single-stage, multistage, and pyramidal types.

The design and operating principle of a single-stage magnetic decoder is illustrated by Fig.89 which shows a circuit of such a decoder with two inputs A and B. The decoder itself is built of four ferrite-transistor cells. The paraphase code of a number is read out from a two-digit register consisting of ferrite-transistor flip-flops (Fig.63).

The ferrite-transistor cells of the decoder are built on the basis of /172 the superposed field magnetization and blocking system. The ferrite core of a

cell is wound with six windings: the superposed field-magnetization winding  $w_f$ , the read winding  $w_r$ , the base winding  $w_b$ , the collector winding  $w_k$ , and two inhibit windings  $w_{inh}$ . A direct current flows in the field-magnetization winding which, in the absence of signals in the other windings, maintains the ferrite core in a state of saturation (the point  $M_1$  of the hysteresis loop, cf. Fig.35c). Consequently, superposed magnetization is accomplished on the writing of 1 end if the positive magnetization intensity of the core corresponds to the code "1". Inhibition is accomplished on the readout end, i.e.,

Fig.89 Single-Stage Magnetic Decoder with Two Inputs

Consider the sequence of operation of the decoder in the case where the binary number 10 is recorded in the register ( $A = 1, B = 0$ ). In this case, current will flow through the inhibit windings connected with the zero output of the flip-flop  $T_1$  and with the one output of the flip-flop  $T_2$ . Consequently, current will flow through the inhibit windings of the cores  $FC_1$ ,  $FC_2$ , and  $FC_3$ . The inhibit ampere-turns are superimposed on the field-magnetization ampere-turns; therefore, these cores will be in a state of deeper saturation than the core  $FC_2$  which is acted on only by the field-magnetization ampere-turns. Let us now introduce a readout current pulse  $i_r$ . In the cores  $FC_0$ ,  $FC_1$ ,  $FC_3$  the readout ampere-turns will be unable to overcome the combined action of the inhibit and field-magnetization ampere-turns; therefore, the magnetic state of 173 these cores does not change and no signal appears on their output windings. In contrast, the core  $FC_2$  is switched to the opposite state, the triode  $PT_2$  starts conducting, and the code signal 1 appears at the output  $P_2$ . When the action of the readout ampere-turns has ended, the core  $FC_2$  is returned by the superposed magnetizing field to its original state. Considering the other combinations of

values of the input variables A and B, it is easy to see that the code signal 1 will appear at the decoder output having the same number as the number in the register. For example, for A = 0, B = 0 (number 00) there will be a signal at the input  $P_0$ ; for A = 1, B = 1 (number 11) there will be a signal at the output  $P_3$ ; finally, for A = 0, B = 1 (number 01), there will be a signal at the output  $P_1$ .

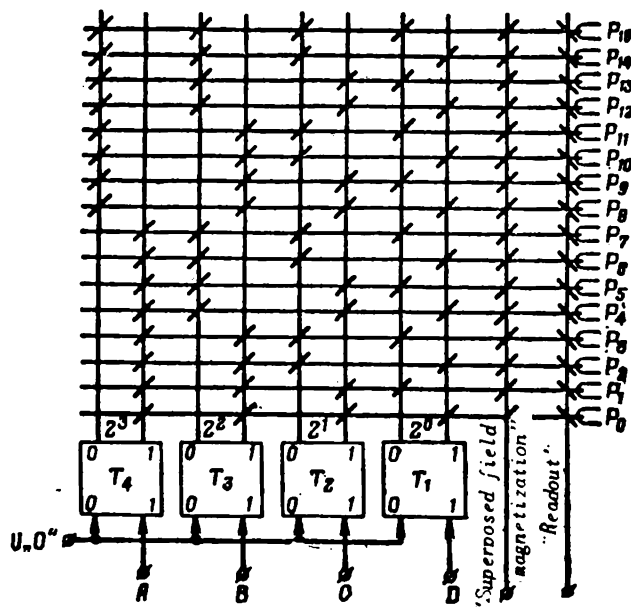


Fig.90 Single-Stage Magnetic Decoder with Four Inputs

Thus, the design principle of a single-stage magnetic decoder of ferrite-transistor cells is that one output bus of the decoder is selected by inhibiting the passage of currents on all buses other than that selected.

Figure 90 is a circuit of a single-stage magnetic decoder with four inputs (16 outputs), built on the above principle. For convenience in representation of the circuit, the ferrite cores of the ferrite-transistor cells are shown by 16 horizontal lines numbered from  $P_0$  to  $P_{15}$ . The windings of the cores are 174 formed by vertical buses over which pass the signals of the paraphase code of the number read out from a four-digit flip-flop register. If the vertical bus forms a winding on the core of a ferrite-transistor cell, then at the point of intersection of this bus with the horizontal line denoting the core, a short slanted line is entered. If, conversely, the bus goes around the core, this line is not given. An inclination of the short line to the right indicates that the core has either an inhibit winding if the corresponding vertical bus is connected with the flip-flop of the register, or a field-magnetization winding. An inclination of the short line to the left indicates the presence of a read winding. Thus, on each core of the ferrite-transistor cells of the decoder there are four inhibit windings and four other windings, namely  $w_t$ ,  $w_k$ ,  $w_b$ ,  $w_r$ . The flip-flops of the register, as in the last decoder, are built

according to the circuit shown in Fig.63.

This decoder operates in the same manner as that of Fig.89. If, for example, the number 1101 ( $A = 1, B = 1, C = 0, D = 1$ ) is stored in the register, then the vertical buses connected with the ones outputs of the flip-flops  $T_1, T_3, T_4$ , and with the zeros output of flip-flop  $T_2$ , will be carrying current. Consequently, the cores of all cells of the decoder except the thirteenth, connected to the output  $P_{13}$ , will be subjected to the action of the inhibiting ampere-turns. On the arrival of a readout pulse, only the core of the thirteenth cell will be switched, and the code signal 1 will appear at the output  $P_{13}$ .

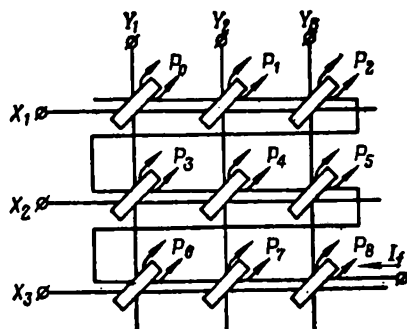


Fig.91 Ferrite Matrix of Magnetic Decoder

It requires  $2^n$  ferrite-transistor cells to build a single-stage magnetic decoder with  $n$  inputs. Such decoders are constructed with a small number of inputs (not over 4 - 6) since an increase in the number of inputs will cause an increase in the number of windings on each core, with consequent difficulty in assembling the circuit. The increase of  $n$  also requires more powerful readout current pulses.

Multistage magnetic decoders. As with the similar diode decoders, the principle of series connection of the stages or matrices, based in this case on cores, is employed.

The cores of the ferrite matrix are arranged in rows, forming a plane rectangular system (Fig.91). Each core has four windings: two inputs, formed by the coordinate buses  $x_i$  and  $y_i$ , one output and one superposed field-magnetization winding formed by a wire passing through all the cores. A DC bias is impressed on the superposed magnetization winding, and returns the magnetic cores to their original state of saturation immediately after they have been 175 switched.

For reversal of magnetization, current pulses are fed to the cores of the matrix over the coordinate buses. The magnitude of the current is so selected that the ampere-turns  $aw_x$  or  $aw_y$  are insufficient to remagnetize the core if the current pulse is fed only to a single input winding. If, however, the pulses arrive simultaneously at both input windings of the core, then the resultant total magnetic field strength will reach the value  $H_n$ , causing the core

to be switched to the other state of saturation; its output winding will then contain a current pulse of definite polarity - the control signal.

Consequently, depending on which of the coordinate buses,  $x_i$  or  $y_i$ , is fed with the current pulses, i.e., which buses are excited, the corresponding core of the matrix at the intersection of the excited buses will be remagnetized and thus produce a control signal in the output winding. The selection of the core, meaning the formation of the signal at the matrix output, is thus produced by the principle of addition of the ampere-turns  $aw_x$  and  $aw_y$ .

Figure 92 shows a two-stage magnetic decoder with four inputs. The four-digit binary number to be decoded is recorded in the flip-flop register. In the flip-flops, made of ferrite-transistor cells, pulse outputs are used. In reading out zero, a current pulse appears on the zero output of the flip-flop; in reading out one, a pulse appears on the one output. Between the ferrite matrices of the first stage (Ia and Ib) and the second stage (II), there are shaping ferrite-transistor cells that amplify the signals arriving from one matrix to the other, thereby preventing their attenuation.

The original state of all cores of the shaping elements corresponds to the remanent magnetic induction  $+B_r$  (code "1"). Each time, after switching of its core, a shaping element is set in the original position by the pulse  $U"1"$  (ferrite-transistor shaping element with superposed magnetization on the write end may be used instead).

The cores of the matrices of the first stage are switched by pulses arriving from the flip-flops of the register, on readout of the stored number, while the cores of the matrices of the second stage are remagnetized by pulses arriving from the matrices of the first stage and amplified by the shaping elements. The latter delay the passage of the signal only by the switching time of the core of the element. A pulse arriving from the shaping element has sufficient power to establish a magnetic field of  $0.5 H_s$  strength in the corresponding core of the following matrix. Let the binary number 1001 be /176 stored in the register. On feeding the pulse  $U"0"$  from the flip-flops  $T_1$  and  $T_4$ , one is read out, and from the flip-flops  $T_2$  and  $T_3$ , zero is read out, i.e., the pulses appear at the ones output of the flip-flops  $T_1$  and  $T_4$  and at the zeros output of the flip-flops  $T_2$  and  $T_3$ . Consequently, the cores  $FC_1$  and  $FC_2$ , which are shown by hatching on Fig.92, are subjected to the full action of the code pulses. As a result of the switching of these cores, pulses arrive in their output windings which, after amplification and shaping, arrive at the matrix of the second stage and result in summated ampere-turns in the core  $FC_3$ . The core  $FC_3$  is switched, and the code signal 1 appears at the output  $P_0$ .

Multistage magnetic decoders using ferrite matrices and operating on /177 the coincidence principle during two current pulses arriving on coordinate buses, have the following basic drawback.

In the output windings of the cores of the matrix affected only by the ampere-turns  $aw_x$  or  $aw_y$  (we will call these half-select cores), there appear noise current pulses due to the deviation of the hysteresis loop from rectangular form. The cores in the same row and the same column as the selected core



are half-selected, i.e., an excited coordinate bus passes through these cores. This decreases the signal-to-noise ratio at the decoder output, which in turn adversely affects the operating conditions of other systems connected with the decoder.

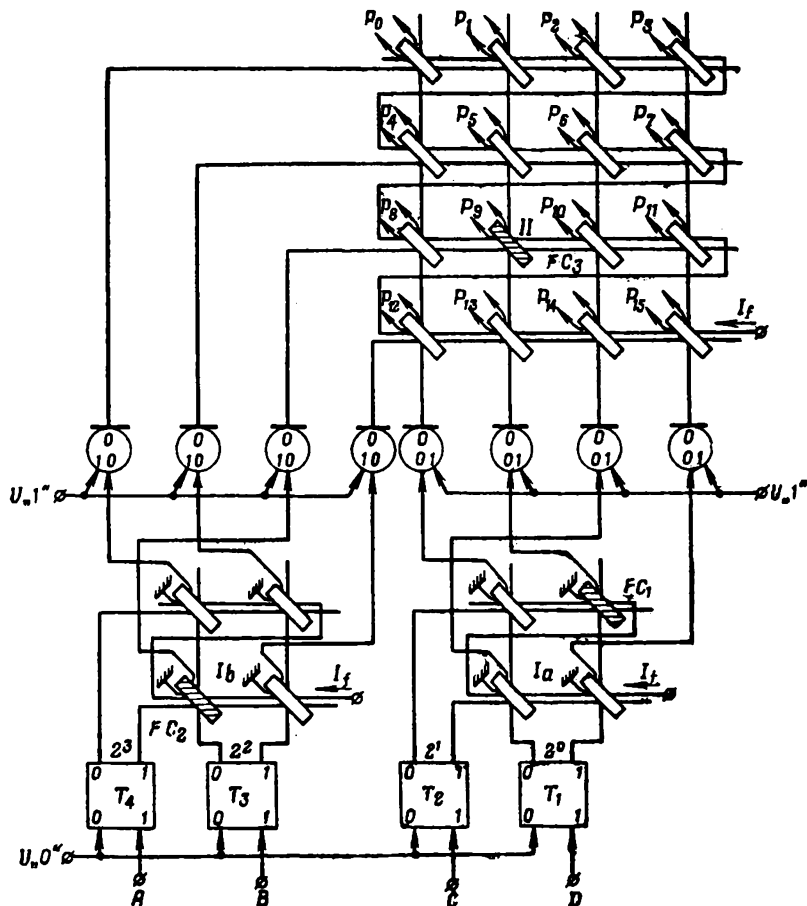


Fig.92 Two-Stage Magnetic Decoder with Four Inputs

To prevent the appearance of noise current due to half-select cores in a matrix, the noise can be compensated by additional compensating cores. To each main (operating) core of a matrix a compensating core is connected, so that the total number of cores is doubled.

Figure 93 illustrates the principle of noise compensation. The write windings of the operating and compensating cores (PC and KC) formed by the coordinate buses  $x$  and  $y$ , are connected in the same direction, while the superposed field-magnetization windings  $w_f$ ,  $w'_f$  and the output windings  $w_{out}$ ,  $w'_{out}$  are connected in tandem. Through the field-magnetization windings flows the direct current  $I_f$ , which determines the state of saturation of both cores. For the operating core, this state is characterized by the point  $A_p$  of the hysteresis loop (Fig.93), and for the compensating core by the point  $A_k$ . If the ampere-turns act on the operating core only on one coordinate, for instance the

ampere-turns  $aw_x$ , a signal whose value depends on the difference  $\Delta B_1 - \Delta B_2$  will appear in its output winding. If the characteristics of the operating and compensating cores are identical, and the number of turns in their windings is the same, then the signal at the output of a half-select core will be very small.

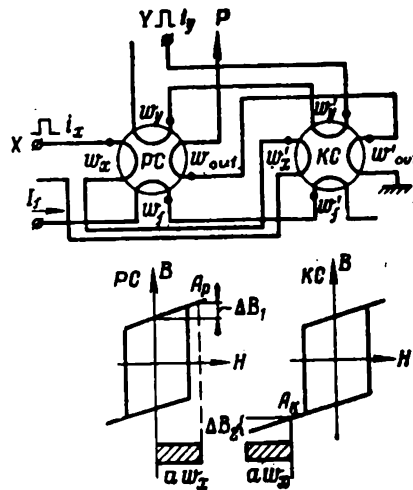
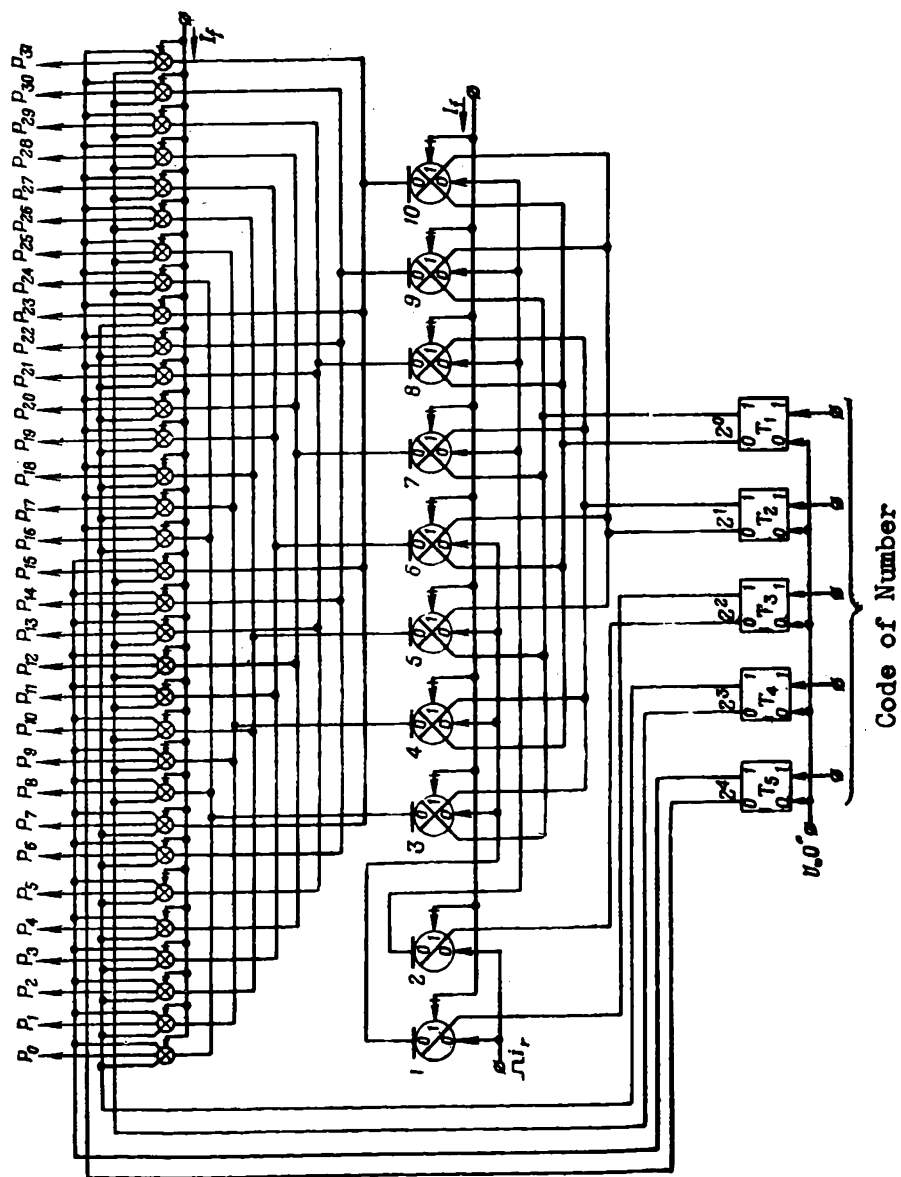


Fig.93 Compensation of Noise in the Output Winding of a Half-Select Core of a Matrix, by Means of a Compensating Core

Among the multistage magnetic decoders, those using ferrite-transistor /178 cells are preferable by reason of their noiseproof feature. In a multistage decoder (Fig.94), the first stage is built up of the ferrite-transistor cells 1 and 2, which are controlled by the flip-flop  $T_3$  of the register, in which the five-digit number to be decoded is stored. The second stage of the decoder consists of cells 3 to 10, and the third stage of the cells whose outputs are denoted by  $P_0 - P_{31}$ .

The flip-flops of the register are designed on the basis of the circuit in Fig.63; the number is read out of the register in the paraphase core. All of the ferrite-transistor cells in the decoder are designed on the superposed field-magnetization system on the write end 1. In contrast to the others, the cores of the cells of the first stage have one inhibit winding each instead of two. In all the cells, the inhibiting ampere-turns act in the opposite sense to the readout ampere-turns. The inhibit windings of the cells in the second stage are fed with signals from the flip-flops  $T_1$  and  $T_2$ , while the inhibit windings of the third-stage cells are fed with signals from the flip-flops  $T_4$  and  $T_5$ .

Let us consider the operation of a decoder with the binary number 11001 (decimal number 25) stored in the register, i.e., when the buses connected with the ones output of the flip-flops  $T_1, T_4$ , and  $T_5$  and the zeros output of the flip-flops  $T_2$  and  $T_3$  are carrying current. Consequently, the current will flow through the inhibit windings of the cores of the cell 2 of the first stage,



**Fig. 94 Multistage Magnetic Decoder using Ferrite-Transistor Cells**

through all cells of the second stage except the cells 4 and 8, and through all cells of the third stage except the cells with outputs from  $P_{24}$  to  $P_{31}$ . On arrival of the current pulse  $i_r$  in the readout windings of the cells 1 and 2 of

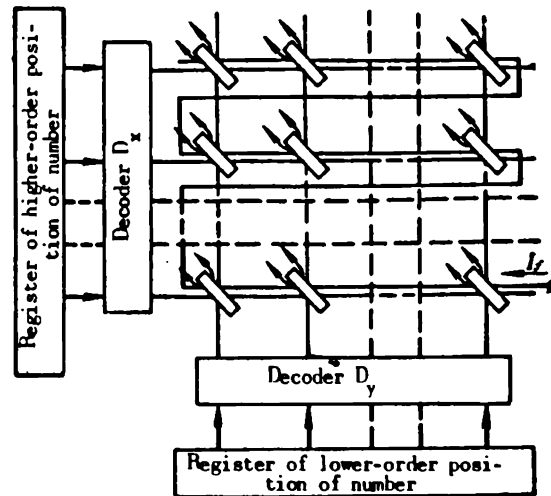


Fig.95 Magnetic Decoder of Composite Type

the first stage, only the core of the cell 1 will be switched to the opposite state, since the readout ampere-turns  $aw_r$  in the core of the cell 2 cannot overcome the combined action of the superposed magnetization ampere-turns  $aw_r$ ,

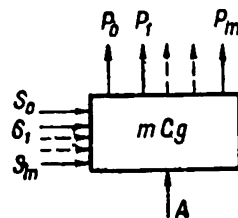


Fig.96 Symbolic Representation of Single-Digit Shifter for Shifting a Number by  $m$  Places

and the inhibiting ampere-turns  $aw_{inh}$ .

As a result of switching of the core of the cell 1, a current pulse will appear at its output, passing into the readout windings of the cells 3, 4, 5, and 6 of the second stage. Under the action of this pulse, only the core of the cell 4 is switched. The current pulse at the output of the cell 4 then passes into the readout windings of the cells with the outputs  $P_1$ ,  $P_9$ ,  $P_{17}$ ,  $P_{25}$ ; as a result, only the core of the cell with the output  $P_{25}$  is remagnetized. The cores of the other cells do not change their magnetic state, since their switching is prevented by the inhibiting ampere-turns. Thus, if the number

11001 (25) is stored in the register, the code signal 1 will appear at the output  $P_{25}$  of the decoder.

The operating speed of a multistage magnetic decoder is determined by the switching time  $\tau$  of a ferrite core and by the number of stages. The time  $t = k\tau$  (where  $k$  is the number of stages in the decoder) elapses, from the time of feeding the readout current pulse  $i_r$  to the time of appearance of a signal at the output.

A multistage decoder using ferrite-transistor cells is characterized by highly reliable operation over a wide temperature range (from  $-60$  to  $+65^\circ\text{C}$ ), and at great deviations of the feed voltage from its rated value (as great as  $\pm 20\%$ ). /180

For a large number of inputs, it is expedient to build a decoder of composite type (Fig.95). The signals arrive at the matrix from the decoders of the first stages  $D_x$  and  $D_y$ , which may be built according to one of the methods given above.

The code of a number to be converted into a control signal is divided into two parts. The codes of the lower-order positions are written into the register of the lower positions of the number, which controls the operation of the decoder  $D_y$ , while the codes of the higher positions are stored in the register of the higher positions, which controls the operation of the decoder  $D_x$ . The control signal appears in the output winding of that core of the matrix which is located at the intersection of the excited buses of the decoders  $D_x$  and  $D_y$ .

## Section 28. Shifters

Devices that perform the operation of shifting the code of a number several places in one direction or the other are called shifters.

A shift is necessary for the multiplication of binary numbers in the natural form, for arithmetic operations on numbers in the normal form (a shift to the left for normalizing a number or a shift to the right for equalizing the exponents), and in other cases. /181

The simplest shifter is the single-digit type; this device shifts a single-digit binary number. Figure 96 is a symbolic representation of such a shifter for a shift of  $m$  places. The inputs  $S_0, S_1, \dots, S_m$  are used to feed the control signals, and the input  $A$  to feed the code of the single-digit numbers to be shifted. Depending on the input to which the control signal is fed, the code of the number will appear on one output  $P_i$  ( $i = 0, 1, \dots, m$ ) or another, i.e., it will be shifted a definite number of places.

The formulas defining the logic of shifter operation are derived for each specific case. For example, the operation of a single-digit shifter serving to shift the number five places to the right is described by the following formulas:

$$P_0 = S_0 A; \quad P_1 = S_1 A; \quad P_2 = S_2 A; \\ P_3 = S_3 A; \quad P_4 = S_4 A; \quad P_5 = S_5 A.$$

In accordance with these formulas we give in Fig.97 a function circuit of a single-digit shifter composed of AND logic elements. The control signal is

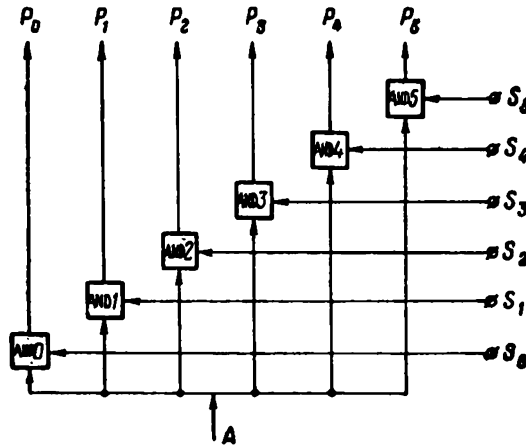


Fig.97 Function Circuit of Single-Digit Shifter for Shifting a Number Five Places to the Right

fed only to the input that will shift the number by the required number of places. If it is necessary to shift a number three places to the right, the

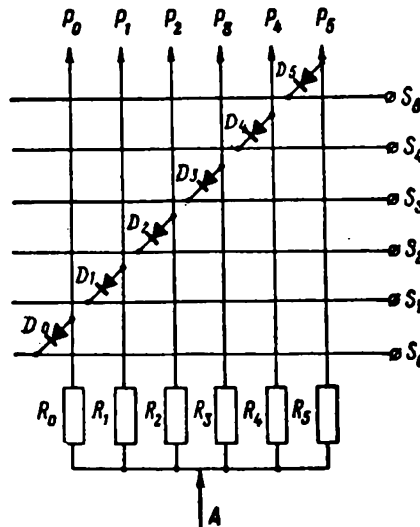


Fig.98 Schematic Diagram of Single-Digit Shifter for Shifting a Number Five Places to the Right

control signal is fed to the input  $S_3$ , causing the code of the number to be passed by the circuit AND-3 to the output.

Figure 98 is a schematic diagram of a single-digit shifter for five places to the right. The shifter is composed of semiconductor diodes. To shift a 182 number, for example, by four places, the high potential is fed to the input  $S_4$ ,

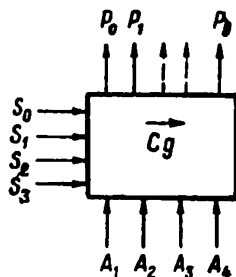


Fig.99 Symbolic Representation of Shifter for Shifting  
a Four-Digit Number to the Right

while the other control buses are under low potential. The code "1" to be shifted is represented by the high potential. Consequently, the diodes  $D_0$ ,  $D_1$ ,  $D_2$ ,  $D_3$ , and  $D_5$  conduct current, and a voltage drop is established across the high-ohmic resistors  $R_0$ ,  $R_1$ ,  $R_2$ ,  $R_3$ , and  $R_5$ , causing low potentials to appear at the outputs  $P_0$ ,  $P_1$ ,  $P_2$ ,  $P_3$ , and  $P_5$ . Since the diode  $D_4$  is cut off by the control signal, no current will flow through the resistor  $R_4$ , and the high potential (code "1") will be transmitted to the output  $P_4$ .

When the low potential representing the code "0" is fed to the input  $A$ , the same potentials will appear at all outputs, regardless of the output to which the control signal had originally been fed.

Consider a more complex shifter (Fig.99) that shifts a four-digit number three places to the right. The binary number  $A_1A_2A_3A_4$  is fed to the input of the shifter. It passes, either not shifted or shifted, one, two, or three places, depending on whether the control signal had been fed to the input  $S_0$ ,  $S_1$ ,  $S_2$ , or  $S_3$ .

The logic of the operation of such a shifter is described by the formulas

$$\begin{aligned}
 P_0 &= S_0A_1; \\
 P_1 &= S_0A_2 + S_1A_1; \\
 P_2 &= S_0A_3 + S_1A_2 + S_2A_1; \\
 P_3 &= S_0A_4 + S_1A_3 + S_2A_2 + S_3A_1; \\
 P_4 &= S_1A_4 + S_2A_3 + S_3A_2; \\
 P_5 &= S_2A_4 + S_3A_3; \\
 P_6 &= S_3A_4.
 \end{aligned}$$

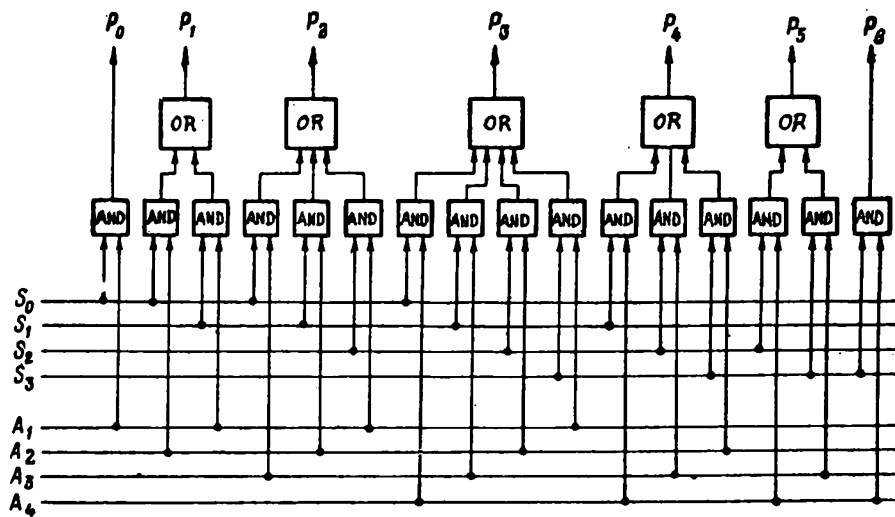


Fig.100 Function Circuit of Shifter for Shifting a Four-Digit Number to the Right

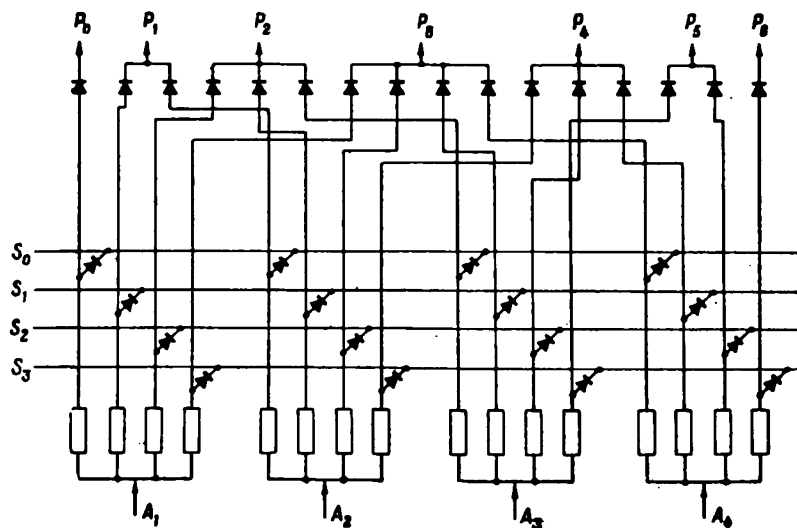


Fig.101 Schematic Diagram of Shifter for Shifting a Four-Digit Number to the Right



Figures 100 and 101 give a function circuit and schematic diagram for /184  
a shifter constructed in accordance with these formulas.

To perform the operation of shifting, shift registers are also used. In contrast to the above shifters, in which a shift by one or several places is performed in a single cycle, the shift register must be fed  $m$  shift pulses in succession in order to shift a number by  $m$  places.

---

## MEMORY UNITS

Section 29. Types of Memory Units and their Main Characteristics

Memory units (MU) of electronic digital computers are designed to receive, store, and put out numbers and instructions to other units of the computer. The MU stores the original data, the intermediate and final results of calculations, the constant variables, and the program instructions.

Up to now no memory unit that would completely satisfy both basic requirements for a memory unit has ever been built. These requirements are large storage capacity and adequate speed. In this respect, a digital computer as a rule has more than one memory unit - two and sometimes even three - which differ in technical characteristics and in operating principle.

Types of memory units. The following types of memory units are distinguished, according to the physical design principle:

- Punch cards and punched tape;
- MU using magnetic tapes, magnetic drums, or magnetic disks;
- MU using delay lines;
- MU using cathode-ray tubes (CRT);
- MU using ferrite cores and ferrite plates;
- MU of diode-transformer type;
- capacitative MU;
- ferroelectric MU.

Memory units based on ferrite cores, on magnetic drums, and magnetic tapes, or on punch cards and punched tapes, are most widely used in the digital computers of today.

According to purpose, the memory units of digital computers are classified into internal (machine memory), external (storage) and intermediate (buffer MU).

Internal MU are distinguished by high speed (a number can be recorded or read out in several microseconds) and by relatively small capacity. In the existing digital computers the usual capacity of an internal MU is 2048 or /186 4096 numbers. Individual numbers are written into, or read out from, an internal MU instead of groups or blocks of numbers as is the case with an external MU. Internal MU operate synchronously with other high-speed units of the computer.

In general-purpose computers, the internal MU in most cases consists only of an operative storage, or a working memory unit (WMU). Special-purpose digital computers frequently have a permanent memory unit (PMU) as well. The working memory only stores the information (original data and program instructions)

required for the solution of a particular problem or for performing the next series of computations. On passage to another problem, the locations of the working memory are cleared and the information required for this problem is stored in them instead. The intermediate and final results of the solution are also recorded in the WMU. The permanent memory stores the quantities that are the same for all problems to be solved on that particular computer. These in-

clude frequently encountered numbers (1, 0,  $\pi$ ,  $\frac{1}{\pi}$ , e, and so on), standard

subroutines, from which the values of common functions (sin, cos, tan,  $e^x$ , and so on) are calculated. The PMU of special-purpose computers usually stores the programs for the solution of problems. The main parameters of a digital computer, especially its speed, depend largely on the characteristics of the internal memory; therefore, a great amount of attention is paid to the development of the MU.

The internal memory units (WMU and PMU) of existing computers are in most cases built of ferrite cores with rectangular hysteresis loop. Ferrite plates are also used, and, far less often, delay lines, cathode-ray tubes, and magnetic drums.

External memory units, on the other hand, are distinguished by a practically unlimited capacity, but are relatively slow: Depending on type, the write or read time ranges from several thousand to several tens of thousands of numbers a minute. When an external MU is addressed, groups of numbers instead of individual numbers are written or read. External MU do not participate directly in the computation. They store the data used in the computer as convenient, at times determined by the program. During the operation of the computer, information is exchanged between the WMU and the external MU.

External storage units are mostly made of magnetic tapes and magnetic drums. Punch cards and punched tapes are also used as information carriers.

Intermediate (buffer) MU are used to coordinate the work of the high-speed working memory with that of the external storage units or with the communication channels, if the electronic digital computer is used in an automated system of information collection and processing and, consequently, is automatically connected to the sources and consumers of information. A magnetic drum is usually used as a buffer MU. /187

Memory units are divided into dynamic and static.

In a dynamic memory unit, the data represented by the signals are in motion, usually periodic, with respect to the medium in which they will be represented. In other words, in a dynamic MU, the numbers are stored by keeping them circulating continuously in some closed loop. The most widely used type of dynamic MU is one based on ultrasonic mercury delay lines. Dynamic MU are used relatively seldom and, in most cases, as internal MU.

In a static memory unit, the data to be stored remain fixed with respect to the storing medium, and only when read out are they converted by a special

device into signals of some type or other. Examples of static MU are units based on cathode-ray tubes, magnetic drums, or magnetic cores. Static MU are most widely used and are employed both as internal memories or external storage units.

According to the character of data storage, memory units may be either periodic or aperiodic.

In periodic memory units, the stored data are cyclically permuted relative to the read or write elements. Such units include MU based on ultrasonic and magnetostrictive delay lines, and MU based on magnetic drums or magnetic tapes.

The main physical characteristics of a periodic memory unit are as follows:

a) Operating period, i.e., the time interval between two successive appearances of the same memory location at the point where its contents are read out by special devices; the operating period is defined by the length of the path over which the signals representing the stored data are displaced in the unit, and by the speed with which these signals move with respect to the read and write elements;

b) Number of digits that can be simultaneously stored in the memory unit; this quantity is determined by the length of the path of displacement of the signals representing the digits, and by the length of the signals themselves.

As an example, let us determine the operating period of an ultrasonic mercury-filled pipe used as memory unit and its length, if 16 binary 40-digit numbers must be recorded in the pipe; the pulse interval is 1  $\mu$ sec and the speed of sound in the storage medium is  $1.46 \times 10^5$  cm/sec.

With these data 640 ( $40 \times 16$ ) codes of the digits of the numbers must be stored in the pipe. The pipe must be laid out for a latency time of 640  $\mu$ sec ( $1 \mu\text{sec} \times 640$ ). This delay is obtained at a length of the mercury-filled pipe of /188

$$L = 640 \cdot 10^{-6} \cdot 1.46 \cdot 10^5 = 93.44 \text{ cm.}$$

In periodic memory units, the time required for reading or writing a number depends directly on the operating period. This time is not constant from case to case, since it includes the waiting time between receipt of a read or write order for any address and arrival of the location with the required address at the read or write elements. The waiting time depends on where these cells with the required address are located at the time the instruction is received. For example, in the ultrasonic mercury delay line memory unit of the BESM computer, the maximum waiting time for a number code is 640  $\mu$ sec.

Dynamic memory units are units of the periodic type.

Periodic MU are sometimes called units without random access, since the waiting time (the time for locating the required cell) is not zero.

In nonperiodic memory units, the read or write time for a number is constant, regardless of the cell of the MU that is required. Examples of nonperiodic units are those based on cathode-ray tubes, magnetic cores, or ferroelectric devices. The speed of such units is determined primarily by the speed at which the information read and write systems operate.

Nonperiodic MU are faster than periodic types. The write or read time of a number is usually a few microseconds, while in periodic MU it is hundreds and even thousands of microseconds.

Memory units of the nonperiodic type are sometimes called units with random access, since the search time is zero. Nonperiodic MU as a rule are used as internal memory units.

Certain characteristics of memory units are used for evaluation and comparison of technical performance.

### Main Characteristics of Memory Units

1. The capacity of a MU is the greatest possible number of number and instruction codes of a definite class that can be simultaneously stored in this unit. This capacity, to a considerable extent, determines the complexity and in some cases also the speed of the unit; for example, in periodic MU, an increase in capacity will also increase the time for writing or reading a number from a cell with a given address. /189

2. The access time to an MU is the time required to read or write the code of one number (or information) at a specified address in the MU. The access time is a characteristic of internal MU.

In the general case, in writing the code of a number, the circulation time  $t_{cir.w}$  in a MU is composed of the search time  $t_s$  for the required cell, the erase time  $t_{er}$  of the old information in that cell, and the time of direct writing  $t_w$  of the number in the cell so selected, i.e.,

$$t_{cir.w} = t_s + t_{er} + t_w$$

The circulation time for readout of a number,  $t_{cir.r}$  is composed of the search time for the required cell, the direct reading time  $t_r$ , and the time for recovery (regeneration) of the number read out,  $t_{reg}$ , i.e.,

$$t_{cir.r} = t_s + t_r + t_{reg}$$

For the existing types of MU it can in practice be considered that

$$t_{cir.w} = t_{cir.r} = t_{cir}$$

In ferrite-core memory units  $t_s = 0$  and in magnetic-drum or magnetic-tape memory units  $t_{reg} = 0$ , i.e., for specific types of MU, certain elements of the time constituting  $t_{cir}$  may be zero.

The circulation time for a MU has a substantial influence on the speed of the entire computer, since the MU must be repeatedly consulted during operation of the computer.

3. Writing or readout speed is the amount of numbers that can be written or read out of the MU in one second. This speed is a characteristic of external and buffer MU, in which groups of numbers are recorded or read out. For present-day MU, the writing speed (or readout speed) may be as high as 1000 numbers a second or more.

4. Duration of information storage is the characteristic defining the time during which a memory unit is able to store information recorded in it, without regeneration. All MU are divided into two groups according to the distortion-free storage time for recorded information.

The first group includes units in which the recorded data can be stored as long as desired. Such MU do not require regeneration of the stored data. They include magnetic-tape and magnetic-drum MU, magnetic-core MU, ferroelectric MU, and capacitor card MU (capacitative MU).

The second group includes MU in which the signals corresponding to the recorded information are attenuated and distorted with the passage of time, such as, for instance, in cathode-ray tube MU, thus requiring regeneration of the recorded data, which considerably increases the complexity of their circuits. In such MU, moreover, the information may be distorted or lost entirely if there are random failures in the power supply during regeneration of the record. /190

5. Economy is expressed by the number of MU elements per stored number, by the life of the component elements of the unit, by the power consumption and power loss during operation, per unit capacity of the MU. To improve economy it is rational to utilize designs without filaments or vacuum tubes, and also a type of MU that does not require regeneration of the information recorded.

6. Reliability of MU operation is one of the most important characteristics of a memory unit. Reliability of operation is largely determined by the operating principle of the memory elements. A memory unit whose elements operate on the qualitative principle (the "yes-no" principle) is most reliable, i.e., when the presence or absence of a signal determines the appearance or absence of a pulse of definite polarity. All the quantitative relations to be found in determining the type of signals taken off the elements of the memory unit decrease the operating reliability. Reliability is also affected by the value and shape of the signals, especially by the steepness of the pulse edges. Reliability is decreased with increasing number of constituent elements of the MU.

7. Size-weight indices of an MU. Depending on the purpose of the digital computer, these indices may be more or less important. While weight and size

are not of fundamental importance for a general-purpose digital computer operating under stationary conditions, they are of great importance for a special-purpose computer operating under field conditions. In all cases, however, it is desirable to use miniature vacuum tubes, semiconductor diodes and triodes, magnetic cores, and other new memory components and elements. This not only improves the size-weight figure of a memory unit but also increases its economy.

8. Sensitivity to the variation in ambient temperature and humidity, to vibration, shock, jarring, acceleration, or to the influence of various physical fields, and the like is of importance. All these indices are highly important for special-purpose digital computers operating under nonstationary conditions.

In addition to the above main characteristics, other criteria for the evaluation of memory units are the possibility of making a new recording on a location occupied by an old recording, the convenience of copying the recorded data (for example the transfer of data from a worn-out magnetic tape to a new one), the convenience of inspection and supervision, etc.

### Section 30. Punch Cards and Punched Tapes

191

Punch cards and punched tapes are used as carriers for the information, the original data, the program, and the results. A punch card and a punched tape are, on the whole, equivalent.

The original information carriers must meet a number of requirements: low cost, small size, long life, repeated re-use, persistence of records during prolonged storage, easy check of writing and correction of incorrect writing, reliable readout, and possibility of assembling original data from individual pieces. Punch cards and punched tapes, on the whole, do satisfy these requirements.

A punch card (Fig.102) is a rectangular sheet of thin flexible cardboard with one corner cut off. The dimensions of a punch card are standard, and for normal operation of a unit, the entry of data on such cards must be rather exact. The corner is cut off in order to easily spot a card that is incorrectly positioned in the card-feed device.

The field of a punch card is divided into 12 horizontal lines or rows and 80 vertical columns. There are also punch cards with 45 columns, but these are not used in Soviet machines. The numbers and instructions whose places are represented by the binary digits 0 and 1, are recorded by punching holes at the points of intersection of the rows and columns. A hole denotes a one in the corresponding place of the number, and the absence of a hole a zero. The numbers are entered row by row on a punch card, i.e., each number has its own row.

The holes are punched on punch cards by a special electromechanical unit, known as the perforator.

The numbers to be reproduced on punch cards and then fed to the internal memory unit of a computer are generally recorded on standard blanks in the decimal or octal number system.

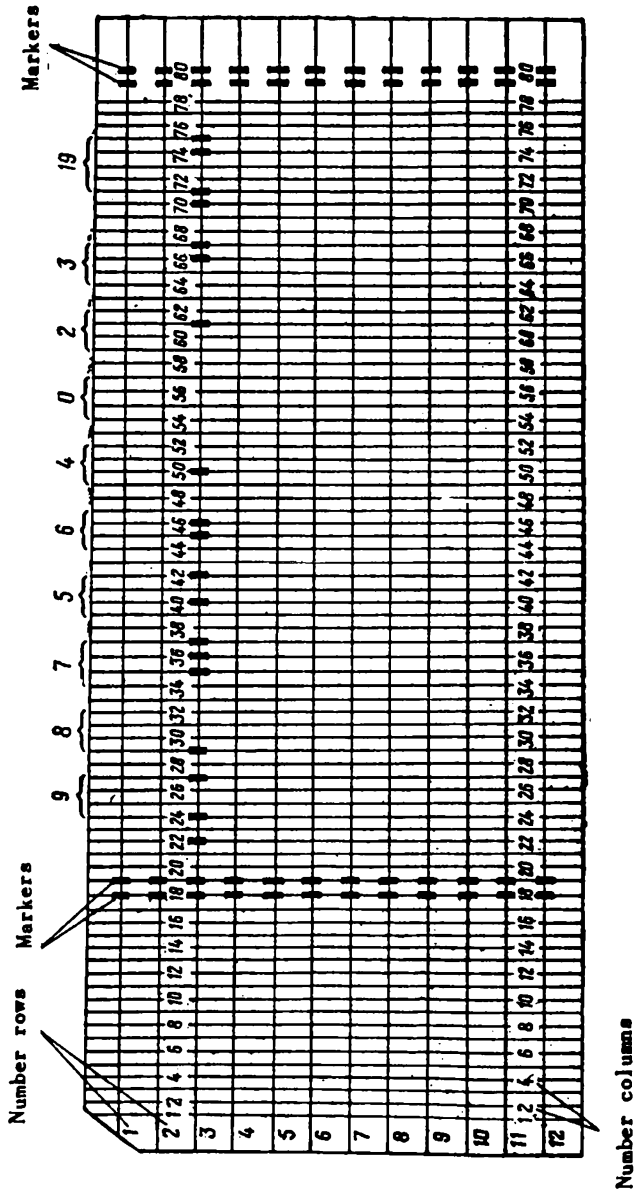


Fig.102 The Punch Card



Each octal digit is represented on the punch card in the form of a three-digit binary number (triad), for which purpose three positions are assigned for each place in the row where the octal number is to be entered. Octal digits on a punch card have the following appearance:

□ □ □ 0	■ □ □ 4
□ □ ■ 1	■ □ ■ 5
□ ■ □ 2	■ ■ □ 6
□ ■ ■ 3	■ ■ ■ 7

The open squares here denote positions without a hole and the black squares, positions with a punched hole. With this representation of octal digits, the number is recorded in the binary system.

Decimal numbers are represented on punch cards in the form of binary-coded decimal numbers. For this purpose, four positions are assigned in the rows 193 of the punch card for each digit of the decimal number, and each decimal digit is entered in the form of a four-digit binary number, or tetrad. The decimal digits appear on the punch card in the following form:

□ □ □ □ 0	□ ■ □ ■ 5
□ □ □ ■ 1	□ ■ ■ □ 6
□ □ ■ □ 2	□ ■ ■ ■ 7
□ □ ■ ■ 3	■ □ □ □ 8
□ ■ □ □ 4	■ □ □ ■ 9

The octal numbers are converted into binary numbers and the decimal numbers into binary-coded decimal numbers, using a special keyboard device; after this, the holes are punched into the card by the perforator.

Figure 102 shows the order of arrangement of the decimal number of  $-0.987564023 \times 10^{-19}$  represented in the normal form, in one row of a punch card, using the binary-coded decimal system. Each digit of the mantissa of the number is represented by the corresponding binary tetrad. There are spaces between the tetrads. To indicate the signs of the mantissa and the exponent, the 22<sup>nd</sup> and 70<sup>th</sup> column respectively are assigned. A minus sign is indicated by a hole of the punch card and a plus sign by absence of a hole. The exponent of the number is represented in binary code. Incomplete utilization of the punch cards is generally due, besides the technical difficulties, to the need for entering additional information, the number of the problem, the number of the punch card, etc. The first 15 columns of the card are used for entering the additional information.

Punch cards are prepared in packs or stacks. The stack of punch cards is placed in the card hopper of the input unit. The punch cards are gripped consecutively by the card gripper and are sensed by a system of contact brushes. According to whether a hole is present or absent at a given location of the card, the contact is either closed or not closed. This contact sends an electrical signal into the computer, and under its action the code "1" is written into the computer memory.

Besides the electromechanical method of reading out information from punch cards, by sensing the punches by contact brushes, the method of readout by means of photocells or photodiodes is also used.

Advantages of punch cards include convenience of their permutation, i.e., of changing the order of the data input to the computer, and of practically unlimited capacity; their disadvantage is the slow readout of information which is limited by the speed of mechanical displacement. The reading speed, when using photodiodes, is considerably greater, up to 180 numbers a second.

Punched tape is a heavy paper or cellulose tape on which, as on the punch card, the numbers are represented by hole systems. /194

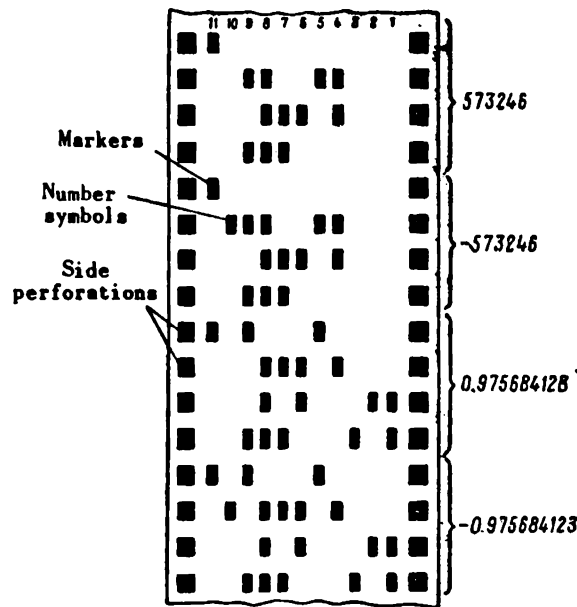


Fig.103 Recording of Octal and Decimal Numbers on the Punched Tape of the "Ural-1" Computer

The codes of numbers can be read off a punched tape electromechanically or photoelectrically. In the latter case, there is a special readout mechanism in the punched-tape storage system. This readout system is based on the utilization of photodiodes. When the opaque tape with the recorded data is passed over it, a narrow beam of electric light is incident from one side. On the other side of the tape is the photodiode system. The photodiodes, on receiving the light pulses passing through the holes in the moving punched tape, produce electrical pulses. By means of these pulses, after amplification, the data are recorded in the internal memory unit of the computer. With the photoelectric method the speed of reading numbers is higher than in the mechanical method and may reach 200 numbers a second.

Let us consider the procedure of recording the codes of numbers on punched tape from the example of the perforated tape used on the Soviet digit com-

puter "Ural-1". This computer uses a blackened celluloid punched tape on a base of ordinary motion-picture film. The material is read out of it for re-writing on the magnetic drum by the photoelectric process.

In recording numbers and instructions, 11 longitudinal rows are punched. In Fig.103, which gives examples of the recording of octal and decimal numbers, these rows are numbered; on the punched tape itself, they are not numbered. Each number or instruction is recorded in four transverse rows, each row being between two holes at the edges of the tape (the side perforations are for engaging the tape transport). The binary digits are represented by a hole at the intersection of a row and column (code "1") or by the absence of such a hole (code "0").

On the eleventh row of the punched tape, a marker symbol is punched to /195 separate the numbers, and on the tenth row various symbols are punched such as

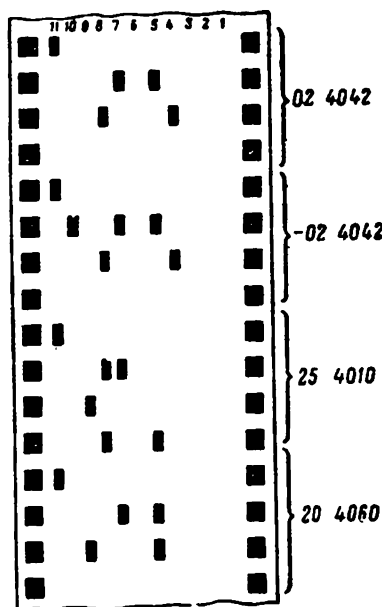


Fig.104 Recording of Order on Punch Tape of "Ural-1" Computer

the minus sign, numbers and instruction symbols, and the area indication "Z" in punching the number of the zone of the punched tape. The numbers themselves, the instructions, and the zone numbers of the tape are punched on the remaining nine rows.

As on punch cards, the decimal numbers are here represented in the binary-coded decimal system, each tetrad being punched into one of the nine rows. The lowest-order digit of the tetrad is punched in the lowest of the four rows of that part of the tape assigned for recording numbers, and the highest-order digit in the top row. For example, the tetrads 0111 and 1001, corresponding to

the decimal numbers 7 and 9, are represented as follows on the tape:

□ 0	■ 1
■ 1	□ 0
■ 1	□ 0
■ 1	■ 1

The tetrad corresponding to the lowest-order digit of the decimal number is punched on the first row and that of the highest-order digit on the ninth.

Octal numbers and instructions are punched into the tape in the binary number system. Each triad is recorded on a separate row, beginning with the ninth. The top row of the part of the tape reserved for write-in of the number is not used in this case.

The instructions are represented in the same way on the tape as the octal numbers. Figure 104 gives an example of the punching of instructions.

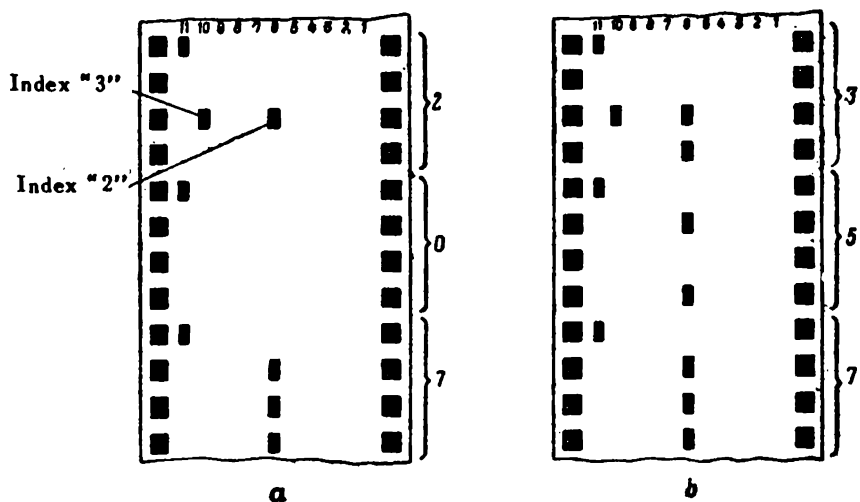


Fig.105 Record of Zone Numbers on Punched Tape  
a - Recording of the 7<sup>th</sup> zone; b - Recording of the 157<sup>th</sup> zone

The numbers of the tape zones are punched on the sixth row, and the zone symbols and the digits 2, respectively, on the tenth and sixth rows. The maximum zone number equals the octal number 177, and in this case the highest-order digit is punched in the "ones" row, together with the zone symbol 2. Figure 105 gives examples of punching for the 7<sup>th</sup> (a) and the 157<sup>th</sup> (b) zones.

Thus the information carriers on punch cards and punched tapes, used for external storage, are static devices. They do not permit a rewriting of the data; information once recorded is permanently preserved. Storage facilities of this type are characterized by practically unlimited capacity but low operating speed. /196

### Section 31. Delay-Line Memory Units

The operating principle of a delay-line memory unit is that information fed to one end of a line in the form of electrical pulses is propagated through it in the form of waves at some definite finite speed and appear at the other end of the line after a certain time. The information taken from the end of a delay line can be transmitted again to its beginning end and can thus continuously circulate in a closed loop. Besides a certain latency time, when information moves along a delay line, distortion of the signals representing this information is also unavoidable. It is therefore necessary for the feedback loop to include amplifying and shaping devices (Fig.106) in order to restore the shape, amplitude, and phase of the signals.

The capacity of a delay-line MU is determined by the number of pulses  $N$  that can be simultaneously located along the delay line

$$N = \frac{Lf}{v},$$

where  $L$  is the length of the line;

$f$  is the code pulse repetition rate;

$v$  is the velocity of propagation of the waves in the medium.

It is obvious from the above formula that it is possible to increase the capacity of a delay-line MU by increasing the length of the delay line, /197

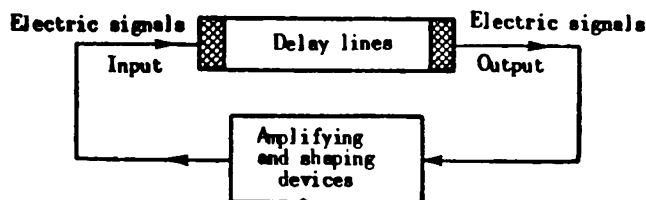


Fig.106 Electroacoustic Delay Line

increasing the code pulse repetition rate, or decreasing the velocity of propagation of the pulses in the medium. An increase in the frequency  $f$ , however, is limited not only by the critical band-pass width of the line, but also by the operating conditions of the other units of the computer (increasing the frequency  $f$  above 1 Mc encounters a number of serious technical difficulties). Lengthening the line will increase the waiting time for the number code and cause attenuation of the signals. Thus the most rational method of building a MU of sufficient capacity is to use a delay line with a low pulse-propagation velocity.

Supersonic delay lines over which mechanical vibrations are propagated at relatively low speed are most widely used. The materials used for the conductor of the mechanical vibrations on ultrasonic delay lines are most often mercury - of the liquids class - and fused quartz or quartzglass, magnesium alloys, and

nickel-alloy wires - of the solids class. The main properties of these materials which are very important for building MU are as follows: velocity of propagation of ultrasound, damping of energy during propagation, temperature dependence of velocity of propagation.

Mercury delay lines were used in the first digital computers.

A mercury delay line is a metal pipe of 30 - 100 cm length and 1 - 2 cm diameter, filled with mercury. The ends of the pipe are closed by quartz piezoelectric transducers, excited in thickness. The piezoelectric transducer, closed at the intake end of the pipe, serves to convert the electrical signals into mechanical ultrasonic vibration. A piezoelectric transducer closed at the outlet end of the pipe serves for the inverse conversion of the ultrasonic vibrations into electrical oscillations. The transfer of the mechanical vibrations from the quartz to the mercury and vice versa takes place with only a small energy loss, a fact that has a favorable effect on the operation of a mercury delay line as a memory unit.

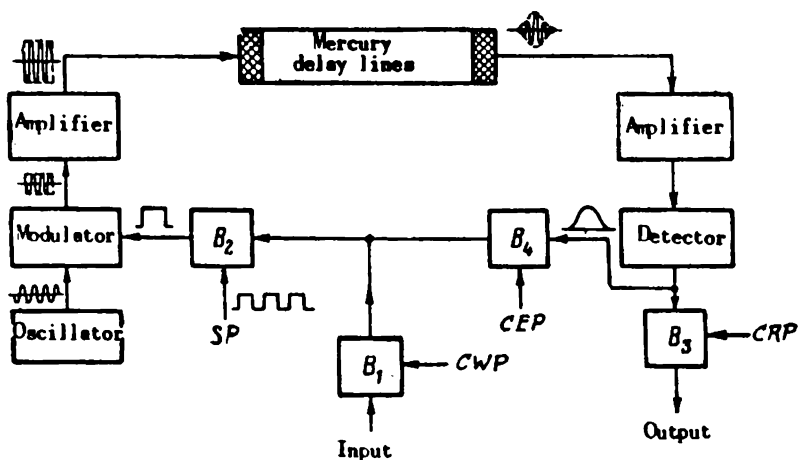


Fig.107 Block Diagram of Memory Unit with a Single Mercury Delay Line

Figure 107 gives a structural diagram of a memory unit with a single ultrasonic mercury pipe. The information for recording is fed to the input of the memory unit in the form of a serial binary code to the gate  $B_1$ , whose second input is fed with the control write pulses CWP. The information is then passed by the gate  $B_2$ , at whose second input the sync pulses SP arrive in continuous sequence. The gate  $B_2$  serves to restore the rectangular shape of the pulses; if the sync pulses have the correct square shape, then the output pulses of the gate will also have sufficiently steep leading and trailing edges, even if the code pulses themselves were "blurred".

The standard pulses taken from the gate  $B_2$  are converted into radio pulses before being fed to the delay line. This is done as follows: Pulses of carrier frequency generated by a high-frequency oscillator arrive at a modulator, where

they are modulated by standard pulses, amplified, and fed to the input of the delay line. Thus "packets" of radio pulses travel from the modulator to the amplifier. Each such "packet" corresponds to the code "1". The use of radio pulses instead of the standard pulses helps better to preserve the shape of the pulses circulating in the closed loop.

The carrier frequency of the oscillator is selected to correspond to the resonance frequency of the quartz crystals, which lies in the range of 5 to 30 Mc. The carrier frequency is usually taken ten times as high as the fundamental frequency of the computer. The fundamental frequency is the repetition rate of the code pulses; in modern digital computers with ultrasonic delay lines, this generally is 1 Mc. Consequently, 10 Mc is taken as the carrier-frequency pulse. /199

By the aid of the quartz piezoelectric crystal at the input of the mercury pipe, the "packets" of HF pulses of electrical voltage are transformed into "packets" of high-frequency ultrasound oscillations. After the reverse conversion, pulses of high-frequency oscillations of electrical voltage appear at the terminal of the pipe, and are then amplified and detected. Detection yields a bell-shape envelope of the pulses. The square shape of the pulses is restored in the gate  $B_2$  by the aid of sync pulses. This constitutes the principle of the circulation of information input to the gate  $B_1$ .

To read out the information stored in a memory unit, the gate  $B_3$  must be opened by the control readout pulses (CRP) at the proper time. The numbers to be read out are taken in serial code and fed to the output register of the memory unit (not shown in the block diagram) whence, when required, they may be put out in parallel code.

In order to erase the code of some number circulating in the closed loop, the gate  $B_4$  must be cut off at the instant at which the code begins to pass through it, by feeding the control erase pulses (CEP). After a certain time, assigned for passage of the number code through the gate  $B_4$ , this gate is again opened (by stopping the feeding of the CEP).

The velocity of propagation of ultrasound in mercury varies with the ambient temperature, so that the latency time of a mercury pipe and thus also its capacity may vary. This may disturb the synchronism of operation of the mercury delay lines with the other units of a computer. To accomplish synchronization, it is either necessary to accurately regulate the temperature by means of a special device or to regulate the frequency of the sync pulse generator in accordance with the temperature fluctuations.

As an example, we give the principal characteristics of the mercury delay-line MU of the BESM series, which uses, as a reserve version, the following:

- Pulse repetition rate in the mercury pipe (fundamental frequency), 1 Mc;
- Carrier frequency from the HF oscillator, 10 Mc;
- Length of one tube, 93.44 cm;
- Number of 40-digit numbers stored in the pipe, 16 (delay time of pipe,

640  $\mu\text{sec}$ ;  
 Maximum waiting time for a number code (operating period of MU),  
 640  $\mu\text{sec}$ ;  
 Number code selection time, 40  $\mu\text{sec}$ ;  
 Number of mercury pipes in the MU, 64 (total capacity, 1024 numbers).

A mercury delay-line MU has the following drawbacks:

/200

The MU circuit must include special devices for restoring the shape and amplitude of the pulses after passing through the mercury pipe.

Auxiliary devices are required to eliminate the temperature dependence of the latency time in the mercury delay line.

The speed is relatively low, due to the periodic system of information retrieval from the unit.

Conversion of parallel number code into serial code and back again is required, if the computer operates on numbers represented in parallel code.

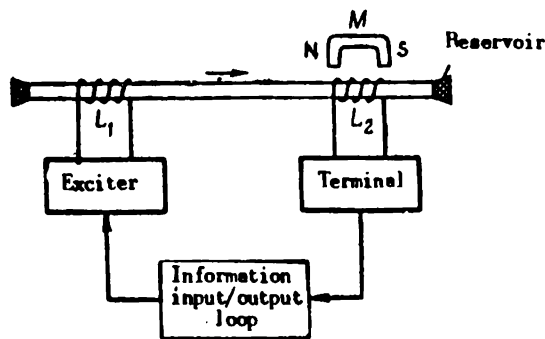


Fig.108 Magnetostriction Delay Line

These disadvantages apply not only to memory units with mercury delay lines but also to units with delay lines of other types.

In addition to the liquid mercury delay lines, solid delay lines are in use. These include the so-called magnetostrictive delay lines, in which the magnetostriction effect is used for transducing the electric energy pulses into ultrasonic vibration energy. This effect consists in a change in shape and size of a given body on magnetization (direct effect), and also in a variation of the magnetic flux in a ferromagnetic material in a magnetic field under mechanical stresses (inverse effect).

A magnetostrictive delay line (Fig.108) consists of a rod of pure nickel with two small coils at its ends, the exciter coil  $L_1$  and the receiver coil  $L_2$ . A current pulse fed to the exciter coil sets up two waves of mechanical stress as a result of the direct magnetostriction effect. These waves are propagated in opposite directions, one of them being damped by the soft-rubber terminal at



the end of the rod, while the other is propagated along the rod at the speed of sound in nickel. This wave is sensed at the opposite end of the delay line by a receiver coil operating on the principle of inverse magnetostriction. In the receiver coil, which is placed in the field of the permanent magnet M, the magnetic flux and induced emf vary under the action of the ultrasonic wave. In this receiver coil the emf is amplified and then used for recirculation or /201 output as the code "1".

Magnetostrictive delay lines are simple and inexpensive. They have a smaller temperature coefficient than other delay lines, i.e., the influence of the temperature on the latency time of the line is less. The shortcoming of such a line is that its frequency passband is narrower than that of the mercury delay line, so that the signals circulating in the closed loop must be of longer duration. Magnetostrictive delay lines do not permit operation at frequencies above 500 - 700 kc.

Electromagnetic delay lines are of three main types: waveguide, long electric line with distributed parameters, and long artificial line with lumped parameters. All these lines have the same shortcoming: They can be effectively utilized in digital computers as memory units only at pulse repetition rates considerably higher than 1 Mc. For this reason, they are not being used in computers.

## Section 32. Memory Units Based on Cathode-Ray Tubes

Digital computers with cathode-ray tubes (crt) are used as the working memory in digital computers. Such MU are very fast, having a circulation time of only a few microseconds with random access to any cell of the MU.

Almost all cumulative crt are based on the so-called potential relief on dielectric screens due to secondary-electron emission. Each digit of a number is represented as a positive or negative electric charge formed on a definite area of the tube screen when it is irradiated by an electron beam. The material of the screen is not a perfect dielectric, and the charge gradually leaks away. This makes it necessary to regenerate the information in order to preserve the charges that represent the data, so that the charges are continually being renewed.

When crt memory units are used, the parallel system of writing and reading the number codes is employed. Each tube is used to store a single digit of all the numbers to be recorded and the number of tubes is determined by the number of digits in the numbers. All the tubes can be simultaneously controlled by a single control circuit.

In all, 512-4096 codes, or even more, can be stored on the screen of one tube.

Three main types of crt with static storage of the charges are used in /202 an electrostatic MU: tubes with a delay or barrier grid, known as potentiometers; tubes with surface redistribution of charges; and tubes of the selec-

tron type with a supporting beam.

All these tubes are constructed on different principles, but all utilize the variation of the potential of a dielectric screen under irradiation by an electron beam.

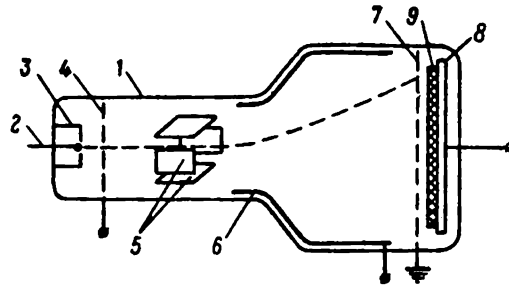


Fig.109 Potentioscope

- 1 - Glass bulb; 2 - Cathode; 3 - Electron gun;
- 4 - Modulating grid; 5 - Deflecting plates;
- 6 - Collector; 7 - Barrier grid; 8 - Signal plate (anode); 9 - Dielectric (screen)

The Soviet BESM and "Strela" computers use potentioscopes for the working memory, while the M-2 computer uses tubes with surface charge redistribution. The latter are widely used in digital computers in other countries, where they are known as Williams tubes.

Let us consider the design and operation of a potentioscope.

The potentioscope (Fig.109) consists of the glass bulb 1, the cathode 2, the electron gun 3, the modulating grid (modulator) 4, the deflecting plates 5, the collector 6, the carrier grid 7, the signal plate (anode) 8, and the dielectric (screen) 9.

The electron gun is used to produce a narrow pencil of electrons (electron beam); the deflecting system, consisting of two pairs of deflecting plates, and the bulb are the same as in the ordinary oscillograph crt.

The electron beam is unblocked and blocked by means of the modulator. The deflecting plates direct the beam to the assigned area of the screen. The screen of the tube is a thin metal plate with a dielectric layer coating on the side facing the collector. The signal plate is usually made of aluminum and the dielectric of aluminum oxide.

The collector is an electrode applied in the form of a coating to the inner surface of the glass bulb. This collector traps the secondary electrons knocked out of the dielectric by the primary electrons and also corrects the reflected scattered primary electrons. It has an output contact to which the potential, necessary for establishing the required electrostatic field potential inside the

tube, is applied.

The barrier grid, which is usually grounded, also traps the secondary <sup>/203</sup> electrons knocked out of the dielectric. This helps to decrease the "seeding" of secondary electrons on the adjacent cells or spots of the dielectric.

The signal plate controls the writing, for which purpose signals of definite polarity are fed to it through its output contact. The pulses are taken off the same plate during reading. Depending on the operating regime, a zero potential (for reading), a positive potential (for writing the code 1) or a negative potential (for writing the code 0) may be impressed on the signal plate.

The dielectric forms the screen struck by the electron beam. On incidence

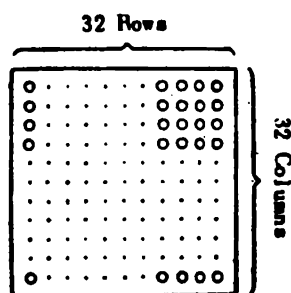


Fig.110 Arrangement of Elementary Cells of a Cathode-Ray Tube Dielectric Memory

of the beam on the various elementary areas of the dielectric surface, these spots become charged to a certain potential, i.e., they play the role of memory spots in the potentiometer. The dielectric screen and the signal plate form the two plates of a capacitor, consisting of a large number of individual elementary capacitors each serving to store one binary digit.

On the surface of the dielectric, the memory spots or cells form a network of vertical and horizontal lines: columns and rows. Figure 110 shows the arrangement of the memory cells on the dielectric of the crt of the BESM computer. The memory cells form 32 rows and 32 columns, making  $32 \times 32 = 1024$  cells. The "Strela" computer has 64 columns and 32 rows, making 2048 memory cells.

Each cell has its own address, consisting of the numbers of its row and column. The potential impressed on the deflecting plate to direct the electron beam to the required memory cell depends on the address of the cell. When a number is recorded, the same potentials are impressed simultaneously on the deflecting cells of all tubes of the memory unit, so that the digits of the number to be recorded are put into the cells with the same row and column numbers in all the tubes. These numbers constitute the address of a number recorded in the memory unit.

During writing or reading of numbers, the electron beam does not sweep the screen continuously. It stops at definite time intervals by feeding pulses of positive polarity to the modulating grid, thus opening the path for the electron beam. Normally the electron beam is blocked by a large negative bias fed to the modulating grid.

Let us consider the physical principles of the operation of a crt with delay grid.

The operation of a potentiometer-type cathode-ray tube is based on the 20 formation of a potential relief on the dielectric surface, as a result of the secondary emission of electrons knocked out of the irradiated cells of the dielectric by the electron beam.

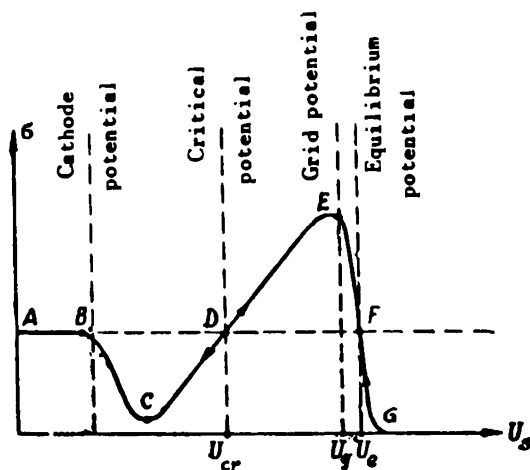


Fig.111 Relation between Coefficient of Secondary Emission and Screen Potential

It has been found that an elementary area of the target - the dielectric screen in a high vacuum - when irradiated by an electron beam, acquires a certain equilibrium potential whose value for a given target material is determined by the secondary emission coefficient  $\sigma$ . This in turn depends on the energy of the electrons bombarding the target, i.e., on the potential of the electric field accelerating the primary electrons.

The secondary emission coefficient  $\sigma$  is the ratio of the number of secondary electrons knocked out of the target by an electron beam to the number of primary electrons impinging on the target. Figure 111 gives a curve showing the relation between the secondary emission coefficient and the screen potential  $U_s$ .

The segment AB of the curve represents the case where the screen potential is below the cathode potential. Here, all the primary electrons are reflected by the screen and are attracted by the collector, so that the secondary emission coefficient is unity. As shown in Fig.111, the equality  $\sigma = 1$  persists at practically all values of the screen potential less than the cathode potential.

With further increase in the screen potential, the secondary emission coefficient drops below unity and decreases almost to zero (the segment BC of the curve). In this case, the screen potential is only slightly higher than the cathode potential. Consequently, the energy of the primary electrons impinging on the screen is insufficient to knock out an appreciable number of secondary electrons. In this case, however, the screen potential is sufficient to retain the primary electrons on the screen itself. A further rise in the screen potential leads to an increase in the energy of the primary electrons and in the number of secondary electrons produced. The secondary emission coefficient thus increases and may become substantially greater than unity (segment CDE of the curve).

Finally, when the screen potential increases still more, there is a sharp fall in the secondary emission coefficient to zero (segment EFG of the curve). This is explained by the fact that, as the positive potential of the screen increases, the energy necessary for detachment of the secondary electrons also increases (the electrons are knocked out of deeper layers) and a retarding force appears, preventing the increase in the number of secondary electrons knocked out. Moreover, if the screen potential is close to the grid potential or even exceeds it, the field established between the screen and the grid will still further increase the energy required for the detachment of secondary electrons. /205

It should be remembered that, on irradiation of the screen, its potential remains constant only for  $\sigma = 1$ . If the number of primary electrons impinging on the screen is greater than the number of secondary electrons knocked out of the screen, i.e., if  $\sigma < 1$ , the screen accumulates a negative charge and its potential decreases continuously. On the other hand, if  $\sigma > 1$ , the screen loses negative charge and its potential increases continuously.

The curve in Fig.111 has three points (B, D, F) at which  $\sigma = 1$  (we exclude the segment AB, which is ordinarily not utilized). However, the point D is a point of unstable equilibrium, since even the smallest random deviation from it either decreases or increases the screen potential. In fact, if under the action of stray factors of some kind the screen potential becomes somewhat greater than the critical  $U_{cr}$ , i.e., the potential corresponding to the unstable point D, then the speed of the primary electrons, and consequently the number of secondary electrons knocked out, will increase. The secondary emission coefficient will become greater than unity. This, in turn, will lead to an increase in the screen potential and to a still greater acceleration of primary electrons. The coefficient  $\sigma$  will continue to increase and so on. By similar reasoning, it can be readily shown that, on any random deviation from the point D to the other side, the screen potential will continuously decrease.

The point F is a point of stable equilibrium, or the equilibrium point, to which corresponds the equilibrium potential  $U_0$ . If, at the beginning of irradiation of some area of the screen, its potential was somewhat greater or somewhat smaller than the equilibrium potential, it will become equal to the equilibrium potential within a short time. This motion toward a stable state is symbolically shown by the arrows. The property of an irradiated area of the screen to pass spontaneously to the equilibrium potential is utilized for

writing data on the screen.

Thus, when an electron beam irradiates a screen with a potential above the cathode potential, there are only two stable states of the screen potential: the potential equal to the cathode potential, and the equilibrium potential. The critical potential separates the two fundamentally different regions of values of the screen potential: If  $U_s < U_{cr}$ , then, on irradiation, the screen will take the cathode potential, but if  $U_s > U_{cr}$ , then on irradiation, the screen will take the equilibrium potential. /206

In the general case, the domain of the stable state into which an irradiated area of the screen is to pass, can be assigned by the variation of three parameters: screen potential, grid potential, and cathode potential. A screen potential  $U_s$  higher than the critical potential is usually employed in potentiometers. In consequence, an area assumes the equilibrium potential on irradiation. The potentials of the cathode and grid are held constant, while the external write pulses are fed to the signal plate. Consequently, the operation of the tube makes use of the mechanism that returns the irradiated part of the screen to the equilibrium state as soon as the screen deviates from the equilibrium potential on account of the external pulses.

In potentiometers, the grid is usually at zero potential, i.e.,  $U_g = 0$ . The use of a grid located in the immediate proximity of the screen (grid-to-screen distance about 0.2 mm) not only decreases the "seeding" or spilling of secondary spurious electrons on the adjoining cells of the screen, but also gives a sharper variation of the secondary emission coefficient on the EFG segment of the curve (Fig.111). For this reason, practically,  $U_g \approx U_s$ .

Let us consider the character of the variation in potential of the irradiated area of the screen when the codes "1" and "0" are written. If no writing takes place, the potential of the signal plate is zero. In writing the code "1", a positive voltage pulse (relative to the grid) is fed to the signal plate; this pulse has the amplitude  $+U$ , which increases the potential of the dielectric and plate. The electron beam, directed toward the area of the screen selected for writing, drops in potential to the equilibrium potential, i.e., to zero. The other elements of the screen are at a potential  $+U$ . After removal of the positive voltage from the signal plate and simultaneous extinction of the electron beam, which is blocked by the modulating grid, the screen potential falls by the value of the amplitude of the write pulse. Here the potential of the area of the dielectric in which the code "1" is written is lower than the zero potential by the value of  $U$ , and the potentials of the other areas are zero. Consequently, after removal of the positive voltage from the signal plate, the elementary capacitor between the irradiated area of the dielectric and the signal plate is negatively charged with respect to the grid.

In writing the code "0", the opposite picture is observed. A negative voltage pulse of amplitude  $-U$  is impressed on the signal plate, decreasing the potential of the screen. The potential of the screen area on which the code "0" is written increases to the equilibrium level under the action of the electron beam. After removal of the negative voltage from the plate and cessation of the action of the electron beam, the screen potential becomes zero, while the

potential of the area in which the code "0" is written is higher than zero /207 by the quantity  $U$ . Consequently, after removal of the negative voltage from the plate, the elementary capacitor between the area of the dielectric that has been irradiated and the signal plate is now charged positively with respect to the grid.

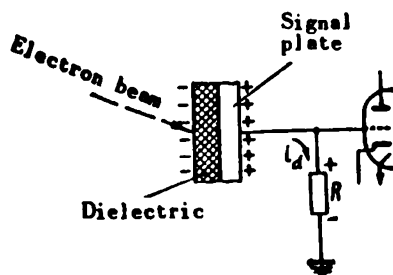


Fig.112 Distribution of Charges in the Elementary Memory Cell of a Potentiometer on Storage of the Code "1"

During reading, no pulses are fed to the signal plate. Depending on the address of the cell of the screen from which the information is to be read, a voltage is impressed on the deflecting plates that will deflect the electron beam to the required spot on the screen. The beam is unblocked by feeding a positive voltage pulse to the modulating grid.

If the code "1" was written in the memory cell, then the charges of the elementary capacitor formed by that part of the dielectric and the signal plate are distributed as shown in Fig.112. At the instant of irradiation, the elementary capacitor is discharged, the negative charge of that part of the dielectric is decreased on account of leakage of the secondary electrons, with a corresponding fall in the positive charge on the signal plate. This latter phenomenon means that the discharge current  $i_d$  passes through the resistor  $R$ . The current  $i_d$  forms the voltage drop  $i_d R$  across the resistor  $R$ ; this voltage drop is applied to the amplifier grid. Consequently, in reading the code "1" a positive pulse arises at the output.

If the code "0" was written in a memory cell, then a negative voltage pulse is taken off the resistor  $R$  on irradiation.

Thus during reading the code stored in a given cell is erased. This means that reading must be accompanied by regeneration or rewriting.

If, after the storage of information on the dielectric screen of the tube, one does not address it, then the charge pattern of the screen can be maintained without significant distortion for several tens of minutes. In fact, the discharge time constant of memory cells, due to galvanic leakage, is usually a few tens of minutes. In the tube, however, there is a spill or "seeding" of the given cell by secondary electrons when information is written into, or read from, neighboring cells. Although this phenomenon is considerably weakened by

the nearness of the grid, the weakening can still be appreciable, even going as far as changing the sign of the signal read out from a given cell, in the event that there is a considerable number of addresses to the adjacent cells. The "spill" varies for various types of address to the cell, being most intense when the code "1" is repeatedly recorded.

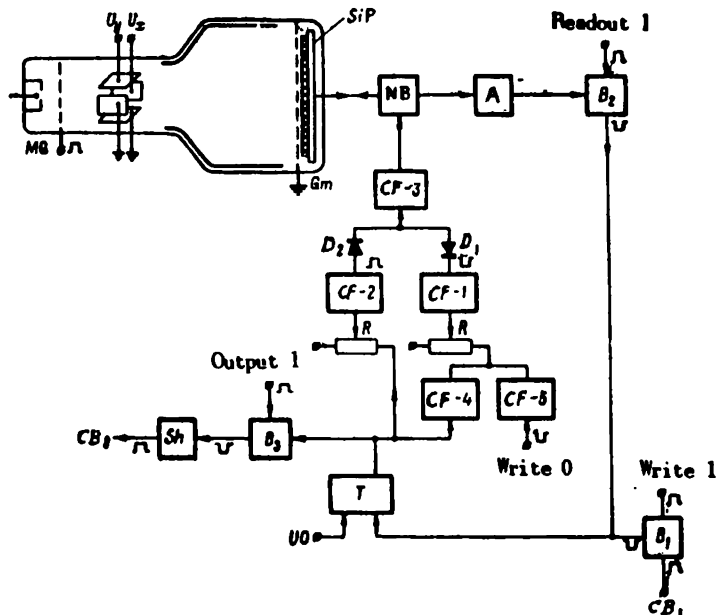


Fig.113 Block Diagram of Electrostatic Memory Unit of a BESM Computer for One Digit of the Numbers to be Stored

Another cause for interference with the recorded information is the /208 "drift" of the voltages feeding the cathode-ray tube, as a result of which the position of the beam on the surface of the dielectric during the reading process is displaced relative to its position during the writing process. Stabilization of the voltages fed to the tube will decrease this phenomena but will not completely eliminate it.

All this necessitates constant regeneration and restoration of the recorded information. This is done in the intervals between addresses to the memory unit, when the memory module is not required either to give out or receive numbers. The electron beam renews the charge pattern of the dielectric every 0.05 sec, whereas an appreciable distortion of the pattern, even in the worst case, takes place only after 0.2 sec.

Figure 113 is a simplified block diagram of the electrostatic memory unit of a BESM computer for a single digit of the stored numbers.

The unit contains the following elements:

1. A potentiometer with the signal plate SiP, the grid  $G_s$ , the modulator MG



which controls the unblocking of the beam, and the deflecting plates along the x and y axes, on which the voltages  $U_x$  and  $U_y$  are impressed.

2. The neutralization bridge NB, which serves to eliminate overloading ("clogging") of the read amplifiers A by the read pulses. /209

In recording or restoring the original writing, a voltage of the order of 10 v is impressed on the signal plate. This voltage, which is several thousand times as great as the useful working signal, also arrives at the input of the read amplifier and there charges its working and stray capacitances. As a result, the amplifier loses sensitivity and, on arrival of the working signal at its input, is no longer able to produce the appropriate pulse at the output. Without special measures, the restoration time for the working state of the amplifier, i.e., the time for discharge of the stray and working capacitances, is considerable (about 200  $\mu$ sec), which limits the operating speed of the memory unit. This time is shortened to 2  $\mu$ sec or less by using a neutralization bridge, which attenuates the write pulse by a factor of over a thousand and brings its amplitude to the same order as the amplitude of the working readout pulse, so that the memory unit now operates on its frequency of 80 kc.

3. The readout amplifier A, serving to amplify the read signals.

4. The read gate  $B_2$ , which selects the signal to be read.

5. The write gate  $B_1$ , on which the signals to be recorded arrive over the code bus.

6. The flip-flop T which serves to record and regenerate the codes "1" and "0".

7. The cathode followers CF-4 and CF-5 with a common load, which are used for recording and regenerating the code "0" and operate as a coincidence circuit.

8. The write cathode followers CF-1, CF-2, CF-3 with the buffer diodes  $D_1$  and  $D_2$ , which serve to separate the write channels for 0 and 1.

9. The code output gate  $B_3$ , designed to put out a pulse on the code bus if the code "1" had been stored in the cell to be read.

10. The shaper Sh to power-amplify the pulse corresponding to the code "1".

Let us consider the operation of the unit during writing, reading, and regeneration of the codes "1" and "0".

In writing a "1", the code pulse "1" arrives over the code bus  $CB_1$  at the input of the gate  $B_1$ . Simultaneously, the second input of this gate is fed with the pulse "write 1". A signal of negative polarity from the output of the gate  $B_1$  arrives at the grid of the right-hand tube of the flip-flop and sets it into the position of the code "1". The high potential taken off the right-hand output of the flip-flop passes through the cathode follower CF-2, the separating

diode  $D_2$ , the cathode follower CF-3 and the neutralization bridge NB to the /210 signal plate of the potentiometer. By this time the electron beam has already been directed to the required cell of the screen: At first, in accordance with the address of the number written into all the tubes of the memory unit, the voltages  $U_x$  and  $U_y$  are fed to the deflecting plates and then, 2.5  $\mu$ sec later, a positive pulse of 6.5  $\mu$ sec duration is fed to the modulating grid, unblocking the electron beam. This is the manner in which the code "1" is recorded.

One microsecond after the end of the pulse, the reset pulse CO is fed to the modulating grid, resetting the flip-flop to its original position 0.

When the code "0" is recorded, no pulse arrives on the code bus, and consequently the flip-flop remains in position 0. The low potential from the right-hand output of the flip-flop travels to the cathode follower CF-4. A negative voltage pulse "write 0" is now fed to this cathode follower, so that a low potential is formed across the common load of the cathode followers CF-4 and CF-5; this pulse moves to the signal plate across the cathode follower CF-1, the diode  $D_1$ , the cathode follower CF-3, and the neutralization bridge NB. Since the beam is already directed to the required cell of the screen, as it is when the code "1" is being recorded, the code "0" is now written.

In reading the recorded code, the beam is directed by the assigned address to a cell or spot of the screen. If the code 1 is recorded in that cell, a positive pulse is formed across the load of the signal plate (neutralization bridge) and, after amplification, is fed to the control grid of the gate  $B_2$ . Simultaneously, the positive pulse "read 1" is fed to the other grid of that gate. The negative pulse arising at the output of the gate  $B_2$  switches the flip-flop to the position of the code 1. The high potential is fed from the output of the flip-flop to the signal plate in order to restore the original writing. This potential also arrives at the gate  $B_2$ , to whose second input the signal "read 1" is fed. The negative pulse from the gate  $B_2$  arrives at the pulse shaper, causing the positive pulse of the code 1 to arrive at the output of the shaper; this pulse is then fed over the code bus  $CB_2$  to the other units of the computer. Within 1  $\mu$ sec after blocking of the beam, the flip-flop is switched to the zero position by the pulse  $U''0$ .

If the code 0 is written in this cell of the screen, then a negative pulse arises across the load of the signal plate and, after amplification, arrives at the control grid of the gate  $B_2$ . Despite the fact that the pulse "read 1" is also fed in this case, there is no negative pulse at the output of the gate  $B_2$ , so that the flip-flop remains in the zero position. Consequently, a low potential is fed to the cathode follower CF-4. Since the negative pulse "write 0" is fed to the cathode follower CF-5, a low potential is obtained /211 across the common load of the cathode followers CF-4 and CF-5 and then, by the above route, arrives at the signal plate to regenerate the original writing of the code 0. At the output of the circuit, in reading out the code 0, there is no signal.

In the case of regeneration of the written code, the memory unit operates as it does for the readout except that the pulse "read 1" is not fed to the input of the gate  $B_2$ , so that no signals arrive on the code bus  $CB_2$ . In all

operating regimes of the unit, the signals "write 0" and U"0" are fed continuously at a definite repetition rate. The signal "read 1" is fed at the same frequency for readout and regeneration.

The principal technical parameters of the potentiometer used in memory units are as follows:

The number of memory cells on the screen of the tube may be as high as a total of 4096.

The time for charging or discharging a memory cell is about 1  $\mu$ sec.

The amplitude of the working signal taken off the signal plate is approximately 0.5 mv.

The sensitivity to deflection of the beam is 0.35 mm/volt.

The maximum number of accesses to a memory cell without regeneration of all the recorded information is 400 - 500.

Potentiometers have several basic drawbacks. The main disadvantages include:

1. To direct the electron beam, voltages of absolutely definite magnitude must be fed to the deflecting plates of the tube. Consequently, the cell of the memory unit is selected by a quantitative criterion (by the magnitude of the deflecting voltages) while all other units of the computer operate on the qualitative principle (presence or absence of a signal), which results in greater reliability of operation. Moreover, the use of a quantitative criterion makes it necessary to stipulate strict requirements as to stability of the power supply.

2. The duration of storage of recorded information is insufficient, so that a recording must frequently be regenerated which is inconvenient during practical operation and complicates the circuit of the memory unit.

3. Readout will erase the information in a memory cell, necessitating its regeneration.

The large dimensions of potentiometers and their complex control circuits makes the entire memory unit bulky and intricate. Potentiometers also have a short life (1000 - 2000 hrs) and are expensive. Nevertheless, potentiometers are used in high-speed computers on account of their high speed.

### Section 33. Magnetic-Drum and Magnetic-Tape Memory Units

/212

Physical principles of magnetic recording. The principle of magnetic writing of the codes of binary digits has lately been used in memory units based on magnetic drums and magnetic tape. Until then, only magnetic sound recording had been in use. The practical experience gained in this field was

utilized to build the memory units of digital computers.

The widespread use of magnetic recording in MU systems has been due to a number of advantageous properties:

Large capacity at relatively small dimensions of the memory unit.

Possibility of repeated use of the recording track by erasing the previous recording.

Unlimited storage of the recorded information without distortion.

Relatively high speed of writing and reproduction of information.

Magnetic recording is based on the property exhibited by many paramagnetic materials of retaining their intensity of magnetization after having been subjected to the action of an external magnetic field.

The ferromagnetic materials used for recording, and therefore called information carriers, may have two senses of magnetization and consequently possess two states of remanence. One of them is characterized by the point  $B_r$  of the hysteresis loop, and the other by the point  $-B_r$  (see Fig.19). The demagnetized state of an information carrier is characterized by the point 0 of the hysteresis loop. Any of these three states may serve as the initial state for signal writing.

The information carrier used is a plastic-binder coating of 10 - 30  $\mu$  in thickness and containing minute particles of ferromagnetic powder in dispersion. This coating is applied on a tape or on a cylindrical base (magnetic drum). The ferromagnetic powder consists of small crystals of ferric oxide  $Fe_2O_3$ .

Electroplated coatings have recently been used as information carriers, specifically a coating of 80% cobalt and 20% nickel deposited onto the surface of the drum in a layer 7 - 15  $\mu$  thick.

The magnetic field by means of which the information is recorded, is established by the magnetic write heads. Magnetic annular heads are mostly used (Fig.114). Such a head consists of an electromagnet with an annular core of thin insulated ferromagnetic plates.

The core has two gaps. The back air gap improves efficiency and lowers the overall remanent magnetization of the core. The front gap, several tens of microns wide, faces the information carrier. The distance between the core and the carrier is of the order of 20 - 50  $\mu$  (in a magnetic-drum MU). /213

When current flows through the winding of the head, a magnetic flux is established in its core and is closed through the gaps of the core and partially through the ferromagnetic coating. After the current ceases to flow, a magnetized region persists in the magnetic layer on account of the remanence. This region represents a magnetic dipole (a small-size magnet). Its polarity depends on the sense of the magnetic flux which, in turn, is determined by the sense of

the current in the winding of the head.

The winding of the magnetic head shown in Fig.114 has a grounded center tap. If an electric signal is fed to the input p, current will flow through the left half-winding of the head, and a magnetic dipole of like polarity, corresponding to the code 1, is formed on the information carrier. When a signal is fed to the input q, a magnetic dipole of opposite polarity, corresponding to the code 0, is formed instead.

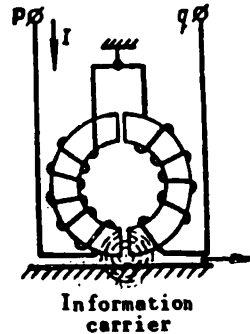


Fig.114 Magnetic Head

If the information carrier is displaced at a certain constant speed with respect to the head and if current of a certain frequency is passed through the windings, then the magnetic dipoles, arranged in tandem, will form a magnetic track.

Longitudinal and transverse recording densities are distinguished. Longitudinal density is the number of code pulses recorded along the carrier (along the magnetic track), which can be entered on 1 mm of its length. By definition:

$$p = \frac{1}{l_{sp}},$$

where  $l_{sp}$  is the distance between the axes of adjacent dipoles, corresponding to the code pulses, i.e., the recording spacing.

The upper limit of longitudinal recording density of code pulses is given by the mutual influence of the fields of adjacent magnetic dipoles, which begins to have an effect at a recording spacing at a smaller length of the dipoles  $l_{sp} < l_d$ , i.e., in overlap recording. The mutual interference of adjacent dipoles of a carrier distorts the shape and decreases the amplitude of the signal to be read out. Nevertheless, overlap recording is still being used since it yields high longitudinal density. The degree of overlap is so arranged that the distortions have no substantial effect on the accuracy of the signals being reproduced. Presently obtainable longitudinal recording density is 10 to 15 pulses/mm.

Transverse recording density is the number of magnetic tracks that can /214

be entered on one full width of the information carrier. The limit in transverse density is given by the thickness of the magnetic head. The distance between the tracks is 1 - 3 mm.

The signals written on the information carrier are reproduced by means of the reproducing or read head. The design of such a head is similar to that of the write head. The same head is sometimes used for both writing and reading. The ferromagnetic coating with the information recorded in the form of magnetic dipoles of one or the other polarity is displaced relative to the read head. Part of the magnetic flux of the elementary magnet (dipole) in the immediate vicinity of the gap of the core of the head, is closed through this core. The time rate-of-change of the magnetic flux caused by the motion of the information carrier induces, in the winding of the read head, an emf proportional to the number of turns of the winding and the time rate-of-change of the flux. This emf is the signal characterizing one bit.

The amplitude of this signal in reading usually does not exceed a few millivolts and depends not only on the speed of the information carrier but also on the recording density. This is due to the interaction of the fields of adjacent magnetic dipoles. With increasing density, the amplitude of the signal decreases.

The process of reproduction causes additional distortions of the shape of the recorded signal and, in particular, increases the duration of the read pulses. This is due to the fact that the magnetic flux in the read head is produced not only by the dipole passing under the front gap of the head at a given instant but also by the adjacent dipoles remote from the gap.

The magnetic coupling between the ferromagnetic layer and the heads is stronger if the poles of the head are in contact with the layer, i.e., if the information is written and read by contact. In this case, pulses of higher frequency can be written and read than if there is a gap between the heads and the ferromagnetic layer. However, the on-contact method of recording can be used only at low linear speeds of the information carrier, for example in a magnetic-tape MU where the tape speed is usually not more than 2.5 m/sec. At higher linear speeds of the information carrier, the on-contact method is inapplicable since the head and the magnetic layer rapidly wear out and break down. In a magnetic-drum MU, a no-contact method for write and read is used, since the linear speed of the drum surface relative to the head may be as high as 100 m/sec.

Magnetic recording consists of three processes: erasing, writing, and reading (reproduction).

Erasing is necessary in order to destroy the data previously recorded <sup>(21)</sup> and return the information carrier to its original state in which new information can be recorded. For erasing, the information carrier is either completely saturated or completely demagnetized. Accordingly, there are two systems of recording data in existence: by remagnetization and by demagnetization.

In the remagnetization write system, the information carrier is first uni-

formly magnetized to saturation in one direction. When recording a zero, the corresponding areas of the information carrier are magnetized in the same direction by current pulses in the winding of the head, so that the state of the

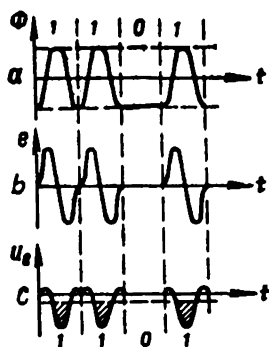


Fig.11.5 Graphs of Variation of  $\Phi$ ,  $e$ , and  $u_o$  for the Remagnetization System of Recording

magnetic coating does not change in these regions. In recording a one, the current will flow through the winding in the opposite sense so that the magnetic coating is magnetized to saturation in the opposite sense, i.e., its state of magnetization is reversed. Consequently, in using a remagnetization system of recording, the information carrier can have only two states of saturation, positive and negative. Such systems are therefore termed two-level recording systems.

In reading, the field of the elementary magnets moving at a definite speed with respect to the read head, is closed through the core. Figure 11.5a is a graph of the time rate-of-change of the magnetic flux in the core. The distance between the adjacent vertical broken lines indicates the time interval required for recording one bit. The figure corresponds to the case where a code sequence 1, 1, 0, 1 passes the head.

The emf induced in the winding of the read head varies in direct proportion to the time rate-of-change of the magnetic flux (Fig.11.5b). The curve  $e = f(t)$  is steepest in the region where the flux  $\Phi$  is close to the maximum. From the winding of the head the voltage travels to an amplifier in which one of the interstage connections contains an RC filter with a small time constant. As a result of differentiation at the output, the curve  $u_o$  shown in Fig.11.5c is obtained. A clipper clips the unhatched part of the curve so that only pulses corresponding to the code "1" remain at the output.

Complete saturation of the information carrier is effected by the permanent magnetic field produced by the erase head, which latter may be an electromagnet with its winding fed by DC or else a permanent magnet.

This recording system is more reliable than a demagnetization system /216 and operates at higher speed. In addition, such a system can use the same head for writing, reading, and erasing. This explains the predominant use of the

remagnetization recording system.

In a demagnetization recording system, the information carrier may be either in two states of magnetization (zero and positive or zero and negative) or three states (zero, positive, and negative). In the former case, the magnetized area of the information carrier corresponds to the code 1, and the unmagnetized area to the code 0.

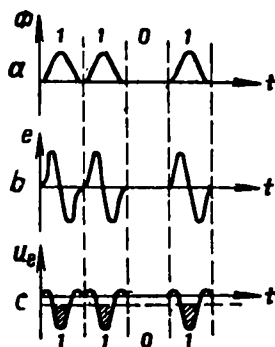


Fig.116 Graphs of Variation of  $\Phi$ ,  $e$ , and  $u_e$  for a Demagnetization System of Recording

Figure 116 shows curves for the magnetic flux, the emf in the windings of the read head, and the pulses of code 1 at the amplifier output. In the latter case (sometimes called three-level recording), the code 1 is recorded by magnetizing an elementary area of the information carrier in one direction, and the code 0 by magnetizing it in the opposite direction. The demagnetized areas indicate the absence of information at these points. At the output, a pulse of one polarity corresponds to the code 1 and a pulse of the opposite polarity to the code 0. In both cases, the information carrier is first demagnetized.

Demagnetization to erase the information is accomplished by an alternating magnetic field induced by an HF alternating current flowing through the winding of the erase head. The magnetic flux, in the form of damping pulses of high frequency is closed through the gap of the core of the head and partially through the magnetized elementary area of the information carrier. As a result, the elementary area is demagnetized to zero.

The demagnetization system of recording requires more equipment per unit MU capacity. The adjustment of such a unit involves considerable trouble, and the write and read rate is low. The erasing of recorded information here is also much more complicated than in the magnetic-reversal recording system.

These recording systems use the so-called return method of recording. Two adjacent magnetic dipoles, representing the code 1 and located on a single carrier track, are separated either by a demagnetized area (in writing systems with demagnetization) or by an area of opposite magnetic polarity (in writing systems with magnetic reversal). Recording systems with magnetic reversal /217 also use writing without intervals or, as it is also called, nonreturn-to-zero



recording (NRZ). In such nonreturn recording, the magnetic state of the track reverses only on passage from code 1 to code 0, or vice versa. Nonreturn recording gives a greater recording density than return-to-zero recording.

Magnetic-drum memory unit. A magnetic drum is a cylinder made of diamagnetic metal, usually of aluminum or an aluminum alloy, with a ferromagnetic coating.

The drum is usually 10 - 30 cm in diameter, but sometimes as much as 80 cm.

According to the character of the data distribution on the drum surface, we distinguish parallel MU (BESM, M-3, etc.), serial MU and parallel-serial MU ("Ural-1" etc.). In a parallel-operation MU, each magnetic track along the circumference on the lateral surface of the drum is allotted to storage of the digits of one place of all the numbers stored. For this reason, all digits of a single number are positioned on the generatrix of the drum. As many numbers can be stored on a drum as there are magnetic dipoles on a single track over the drum circumference.

If the magnetic drum is used as a serial memory unit, each number is placed serially on one track, digit by digit. Each track has several numbers.

The surface of the magnetic drum of the "Ural-1" computer, which uses the parallel-serial system of data location, is divided into five zones, one of which is free while the others carry numbers and instructions. Each 36-digit number is recorded in all four zones, nine digits in each zone. The number is recorded in a single revolution of the drum in four stages, the nine highest-order locations of the number being recorded first, and the nine lowest-order last.

The capacity of a magnetic drum is determined by the area of its lateral surface and the recording density. The recording density, in turn, is determined, on the one hand, by the mutual spacing of the dipoles on the drum and, on the other hand, by the distance between the axes of adjacent tracks. The recording density along the tracks on drums may be 30 and even 40 symbols per cm, and 5 - 8 tracks can be placed on 1 cm of length of a drum. The maximum capacity of a drum is 1.5 - 2 million bits.

The lateral surface of a drum may be increased in order to increase the /218 capacity, by increasing its diameter or length. The increase in drum diameter is limited by design considerations. Moreover, at a given code pulse repetition rate and a given linear drum speed, the memory access time also increases.

With increasing drum length, the amount of hardware also increases, since more read and write heads are required. A substantial saving in hardware, however, can be effected by building the heads in the form of a block which, traveling along the generatrix of the drum, occupies one of several possible positions. For example, in one of the British computers, the data are recorded on, and read from, 256 tracks of the drum by means of only 16 heads. The heads travel to any of 16 positions within 25  $\mu$ sec. This saves 80% of the MU elements, with an increase of only 10% in the access time.

Consider the process of control of the reading and writing of numbers when the magnetic drum is used as a parallel store. Figure 117 gives a circuit of the control loops for writing and reading information from the drum. To simplify the circuit, we show only one head  $H_1$  with the corresponding elements used for writing and reading one digit of all the numbers to be recorded.

The head  $H_2$  is the read head and is placed above a track with previously recorded sync pulses. To each such pulse corresponds the code of a number written along the generatrix of the drum.

The numbers stored on the drum are serially numbered in increasing order, beginning at the origin of reference. The serial number is its address, which is defined by the number of sync pulses applied to the circumference of the drum, beginning with the origin of reference and ending at the position where the number in question is written. For convenience in representing the windings on the wiring diagram, the magnetic heads are turned 90% relative to their true position.

When a number is read from the drum, its address is fed to the address code bus CBA on the receiving register of the address, which has previously been set to zero position by the pulse  $U^{*}0$ . On rotation of the drum, the sync pulses are read by the head  $H_2$  and are then amplified, shaped, and fed to the pulse counter. The counter counts continuously, regardless of whether there is a writing (reading) on the drum or not. In the address comparator, the codes of the numbers to be recorded in the counter are compared with the address codes in the receiving register. If the codes agree, the address comparator sends a pulse to the coincidence circuits AND-2 and AND-1.

The pulse is passed only by the circuit AND-2 to whose second input the "read" signal is fed. The signal from the AND-2 circuit prepares the AND-5 circuit to transmit the code. The pulse read by the head  $H_1$ , corresponding /219 to the code 1 or 0 of one of the digits of the selected number, is therefore passed, after amplification, by the AND-5 circuit and arrives at the code bus CB.

For writing a number, the circuit operates similarly. The address of the position where the number is to be written is likewise fed to the receiving address register and is compared with the code of the numbers on the counter. When the codes agree, the coincidence circuit AND-1 is unblocked and is fed with the signal "write". Depending on the value of the given digit of the number being recorded, a high-level signal is fed from the flip-flop  $T_1$  to the circuit AND-3 or AND-4. Since a pulse from the coincidence circuit AND-1 arrives simultaneously at these circuits, the signal from the flip-flop passes through one of the windings of the head  $H_1$ . In this case, the system of magnetic reversal recording is used, i.e., in writing the code 1 the elementary area of the magnetic coating of the drum is magnetized in one sense, while in recording the code 0 it is magnetized in the opposite sense. To obtain the currents of opposite senses, the head is provided with two windings having a /220 grounded center tap.

In recording new information on a magnetic drum, it is not necessary to

erase the old data since the write pulse, which is of rather high value, will magnetize the magnetic coating in one sense or the other, regardless of what was recorded on it before.

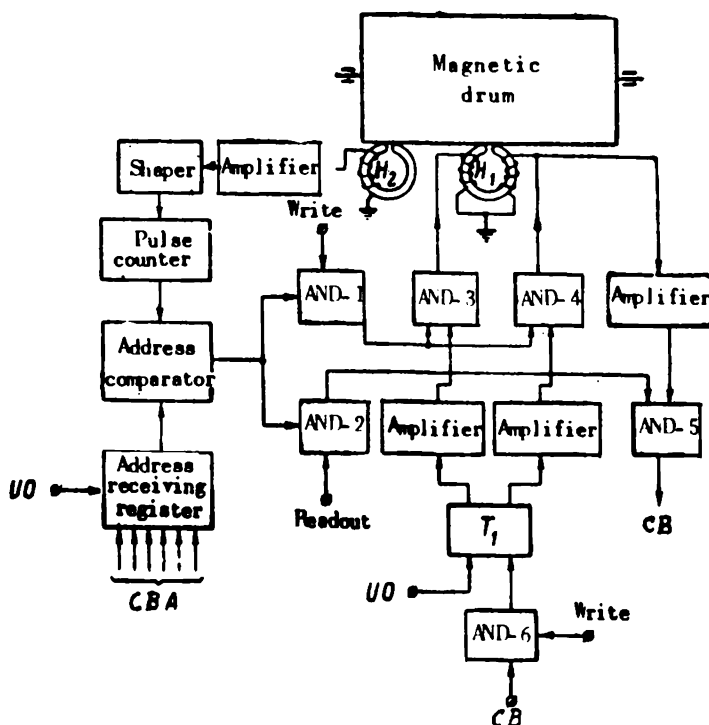


Fig.117 Block Diagram of Magnetic Drum Memory Unit with Parallel System of Data Positioning on the Drum Surface

Figure 118 is a circuit diagram of the write and read amplifier of the magnetic-drum memory of the M-3 computer. The magnetic head has a write winding and a read winding which are, respectively, connected to the write amplifier (6N8 tube) or the read amplifier (6P9 and 6Zh4 tubes) to the flip-flop of the appropriate digit of the register in the arithmetic unit.

Voltage from the cathode followers of the flip-flop in the AU register is fed to the grid of each triode of the tube 6N8. In addition, a positive write pulse arrives at both grids of the write amplifier. The circuit parameters and the pulse amplitude are so selected that the pulse will pass to one triode or the other, according to the state of the flip-flop of the AU register. If, for example, the flip-flop is in state 1, then a high potential (about 200 v) is fed to the grid of the right-hand triodes of 6N8, and a low potential (95 to 100 v) to the grid of the left-hand triode. The write pulse passes through the right-hand triode of 6N8.

The plate circuit of both triodes of the write amplifier includes a push-pull stepdown transformer  $Tr_1$ , with a secondary winding directly connected to

the write winding of the magnetic head. When the code 1 is being recorded, current flows in one direction in the write winding of the magnetic head and /221 in the opposite direction to write the code 0.

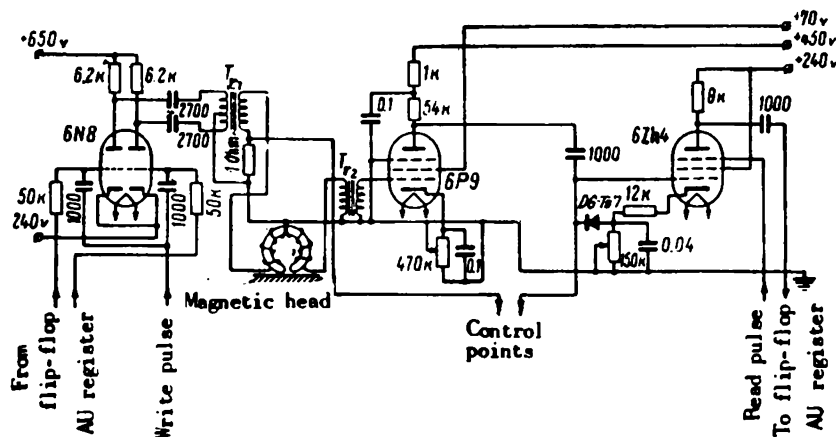


Fig.118 Circuit Diagram of Write and Read Amplifier in the Magnetic-Drum MU of the M-3 Computer

The read winding of the magnetic head is connected to the stepup transformer  $Tr_2$ , in the grid circuit of the first tube of the two-tube read amplifier. The tube 6Zh4 of this amplifier also performs the functions of a gate: at the time of reading and transmission of data from the MU to the AU, a positive pulse is fed to its pentode grid. The pulse from the plate load of the 6Zh4 tube passes across a capacitance to the grid of the flip-flop of the AU register, setting it in position 1 when one is being read.

Magnetic-drum memory units with parallel reading and writing of the digits of a number are faster than serial reading and recording systems, but their design is more complicated since they have a larger number of elements per unit capacity. Memory units with serial reading are slower but are more economical. However, the difference in speed is not very great. This is due to the fact that, in manufacturing and adjusting a module with a large number of magnetic heads (for the system of parallel reading), it is difficult to obtain small deviations of the heads from the common straight line. As a result, the density at which information can be recorded on a drum with parallel reading is somewhat less than on a drum with serial reading. Thus the recording density on the circumference of the magnetic drum of the BESM computer (with parallel reading) is 2 bits/mm.

In this connection, the mixed parallel-serial principle of information positioning on a drum, and the corresponding parallel-serial principle of reading and writing numbers is being used ever more widely today. By comparison with the parallel reading system, it saves a considerable amount of hardware, while the difference in speed is not excessive.

We note the following favorable features of magnetic-drum memory units:

- Large capacity;
- simple operating principle;
- no distortion of information on reading;
- prolonged storage of information without regeneration.

The principal drawbacks of such units are due to their mechanical parts moving at high speed. For example, the stability of the write and read frequency depends on the constant rotary speed of the drum. The reliability of writing and reading depends on the accuracy of the gap between the magnetic coating of the drum and the cores of the heads. The accuracy of manufacture of the drum is also highly important, since manufacturing errors may lead to /222 the appearance of spurious signals.

A substantial shortcoming of magnetic drums for periodic memory units is the impossibility of random access and recording of information. The access time to a magnetic drum MU depends on the waiting time of the number code, which is variable and depends on the rotary speed of the drum and on its dimensions.

Owing to the relatively long access time of a magnetic-drum MU, it cannot be used as an internal memory unit in high-speed computers. The main internal memory units in such computers are faster (cathode-ray tube MU or magnetic-core MU).

The magnetic-tape memory unit. Magnetic tapes are widely used in magnetic recording MU as information carriers, especially as large-capacity external storage, in almost all digital computers.

Magnetic tape consists of a flexible base of acetyl cellulose, polyvinyl chloride, or similar substances, coated with a lacquer film containing 30 to 45 vol.% of a ferromagnetic powder. Tapes of various width are used for external stores, from 6.35 mm with one to three tracks to 125 mm with 50 tracks. Magnetic tape of 17.5 and 35 mm width, with the parallel-serial method of information positioning, is in wide use.

Narrow magnetic tapes are used in serial-type stores. This type of tape has 2 - 3 tracks. If there are three tracks, then one of them is used for the main information, i.e., the numbers and instructions, and the other two are used for auxiliary information such as the numerals of the groups of numbers and the sync pulses. If the tape has two tracks, then the one that takes the sync pulses is used not only to assign the writing and reading rate but also to determine the address of the numbers.

Wide magnetic tapes are used in parallel stores, in which all digits of a number are simultaneously recorded or read. The number of tracks for the basic information is determined by the number of digits in the numbers to be stored. Usually there are marker pulses at the ends of each line used for writing the code of a number. These pulses are formed along the edges of the tape by two tracks. The coincidence of the marker signals indicates that there is no mis-

alignment of the tape and signals the end of the reading of the number code /223 of the line in all the tracks. An example of a memory unit using wide magnetic tapes is the external storage of the "Strela" computer.

The access time to a store using magnetic tape is composed of three main components: the time for starting the tape-transport mechanism, the waiting time for the required group or block of number codes to arrive, and the time for writing and reading the codes.

A magnetic tape is normally in a position of rest or at standstill. The tape transport starts on the instructions to write or read from the tape. Before start of the writing, a certain time will elapse until the tape has reached its normal speed. The minimum starting time for the tape transport used in the BESM and "Strela" computers is 50 - 100 msec, as a result of the considerable inertia of the reels on which the tape is wound. If the start-up time is decreased, the tape may tear since the force applied to the guide reel is transmitted through the tape.

The waiting time for the required group of number codes to reach the head is reduced to zero if there is a preliminary search mechanism to find the zone of the tape where it is located. This device searches for the required zone while the computer is performing other operations.

The write and read time is determined by the recording density, the data positioning system, and the tape speed. For example, in the M-2 computer, at a tape speed of 0.4 m/sec, the read rate is about 1200 bits/sec, and in the "Ural-1" computer about 2700 bits/sec. A higher tape reading rate is attained in the American NORK computer, about 70,000 bits/sec at a tape speed of 4 m/sec. In units with parallel data positioning, the read time is shorter than in units using the serial principle. However, for the same reasons as in magnetic-drum MU, the time difference is not very great. The recording density on a wide tape is appreciably less than on a narrow tape. Thus, in the "Strela" which uses a wide tape, the recording density is about 1.4 bit/mm, and in the BESM which uses a narrow tape, it is 8 bits/mm.

The parallel-serial principle of data positioning on the magnetic tape is finding increased application today (for example in the "Ural-1").

Let us consider the positioning of information on the magnetic tape of the BESM-2, and the control during writing and reading. The magnetic-tape MU of the BESM-2 uses a narrow tape (6.35 mm) with only two tracks (Fig.119): the sync pulse track SP and the code track on which the numbers, instructions, /224 and tape zone numbers are recorded. Writing and reading of the sync pulses is accomplished by the magnetic head  $H_2$ , while the code pulses are written and read by the head  $H_1$ . The sync and code pulses are recorded simultaneously, one sync pulse corresponding to each code pulse. The head  $H_0$  is used to erase information. The method of on-contact recording used has intervals between two levels with preliminary uniform magnetization of the information carrier.

Information is exchanged between the magnetic tape MU and the working memory of the BESM-2 during operation of the computer by groups of numbers,

the transfer being accomplished through the arithmetic unit. The number groups or blocks to be written or read are located in the same zone on the magnetic tape. The zone number is recorded on the code track at the beginning of writing the number group.

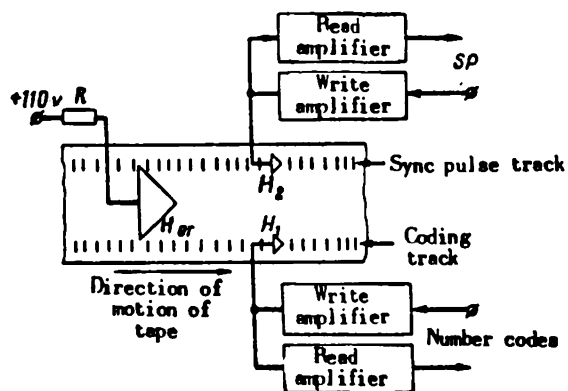


Fig.119 Positioning of Information on the Magnetic Tape of the BESM-2 Computer

Figure 120 gives the control circuit of the process of writing or reading on or from the MU. The circuit operates in the following sequence for reading: The code of the zone number indicated in the address part of the instruction to turn to the magnetic-tape MU is fed to the register  $Re_1$  of the write-read control circuit. The static outputs of the flip-flop in the register  $Re_1$  are coupled with the comparator circuit.

When the magnetic tape moves, the code pulses to be read arrive successively at the input of the shift register  $Re_2$  and are moved into it by sync pulses acting as shift pulses. The first number to be read by the head  $H_1$  is the zone number, for which six places are allotted on the tape. The code of the zone number is fed to the register  $Re_2$ , whose parallel output is likewise coupled to the comparator circuit. If the specified instruction to turn to the MU of the zone number does not agree with what has been read, no control pulse will appear at the output of the comparator circuit. In this case, the reading of the numbers following the zone number by the head  $H_1$  will be blocked. On approaching the head of the next zone, the circuit will operate similarly. 225

If the specified and read zone numbers agree, then the comparator circuit will emit the control pulse  $CP_1$ , which reswitches the control circuits of the memory unit in such a way that the number codes of the specified tape zone to be read will be directed to the arithmetic unit, and then to the working memory of the computer. The control pulse  $CP_2$ , which appears at the output of the AND element, performs the same functions.

The counter  $Co_1$  is switched to zero position each time the number of sync pulses arriving at its input becomes equal to the number of places in the representation of the tape zone number, i.e., it is returned to zero by every

sixth pulse. Consequently, when the codes of the assigned and read zone numbers agree, signals arrive at both inputs of the AND coincidence circuit, and the pulse  $CP_2$  is generated.

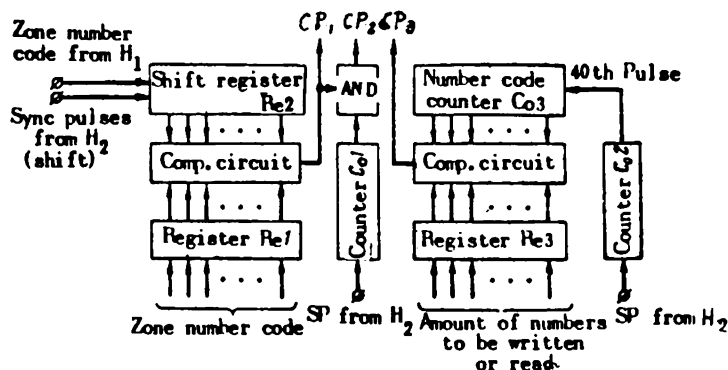


Fig.120 Block Diagram of Write and Read Control in the Magnetic-Tape Memory Unit of the BESM-2 Computer

The sync pulses are also utilized to check the amount of numbers to be copied from the tape into the working memory of the computer. With this object, from the time of coincidence of the specified and read zone numbers, sync pulses begin to arrive at the counter  $Co_2$ . This counter is returned to the zero position and generates a pulse at the output each time the number of pulses arriving at its input corresponds to the number of digits of one number (in the BESM-2, 40 sync pulses correspond to one 39-digit number). Consequently, a pulse will appear at the output of the  $Co_2$  counter every 40<sup>th</sup> sync pulse, indicating that one number has now been read.

The pulses appearing at the output of the  $Co_2$  are counted by the counter  $Co_3$ . The code denoting the amount of numbers to be read is recorded on the register  $Re_3$ . When the codes of the numbers recorded in  $Re_3$  and  $Co_3$  agree, 226 the comparator circuit emits a control pulse  $CP_3$  which stops the counting of the numbers from the tape.

The recording of numbers on the tape is controlled in a similar way.

Magnetic-tape memory units have the same advantages and disadvantages as magnetic-drum units. It must only be noted that the capacity of a magnetic-tape MU is practically unlimited, since the number of tapes for a given computer, which are stored in separate cabinets, may be very great. In addition, the tape access time is far longer than the access time to a magnetic drum.

#### Section 34. Matrix-Type Magnetic Working Memory Units

The use of magnetic cores with a rectangular hysteresis loop is exceptionally widespread today. Such cores are used to build the memory units of digital computers. It is characteristic that magnetic-core MU are more and



more displacing other types of MU. This is due to a number of favorable features of a memory cell with magnetic cores: It can store recorded information without regeneration and without power consumption, for an indefinite time; its service life is long, and its operating reliability is high. It uses only a small amount of power in writing and reading of information. It is small and its cost is low. It is also fast (the access time to such a MU is only several microseconds).

However, such MU also have their disadvantages. Above all, the information recorded on the cores can be read only during the transient accompanying its magnetic reversal. In most cases this erases the recorded information. There is also frequent trouble in matching the current-magnetic elements with vacuum-tube potential elements. Finally, ferrite cores have poor temperature properties.

Working MU as well as permanent MU for storing constant quantities are built of magnetic cores with rectangular hysteresis loops.

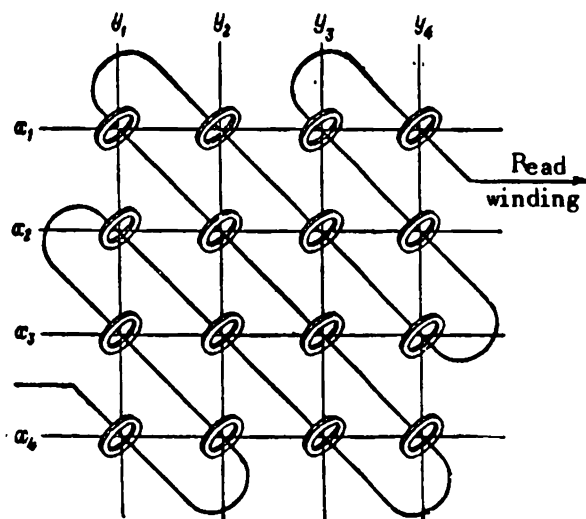


Fig.121 Matrix-Type Ferrite-Matrix of MWMU

A modern magnetic working memory unit (MWMU) of parallel action uses hundreds of thousands of magnetic cores. The methods of positioning the cores and the circuits connecting them are so selected as to decrease the amount of control equipment required and to obtain reliable operation of the memory unit. MWMU of matrix type, and direct-access MWMU or type Z MWMU have found practical use in digital computers.

A matrix-type MWMU uses the matrix method of arranging the cores referring to the same digit of the numbers stored. Let us consider the design principles of matrix circuits, each of which is essentially a MWMU circuit for single-digit numbers.

The coincident current circuit. The circuit of a ferrite matrix of coincident currents used for storing the digits of one place of the numbers to be stored is given in Fig.121. Each core of the matrix has three windings and is used to store one digit of a binary number. The windings are usually of the single-turn type, made of a wire passed through the cores of the matrix. The windings (coordinate or address buses)  $x$  and  $y$  are used to select the corresponding core in writing or reading, and the wire passing through all the cores is the common output winding of the matrix, or the read windings.

Here, as in the ferrite matrix used in the magnetic decoder (see Fig.91), the principle of time coincidence of the currents  $I_x$  and  $I_y$  is used. The information is recorded by the simultaneous driving current pulses of  $I_s/2$  amplitude along one of the  $x$  wires and along one of the  $y$  wires. The selected core, in which the code 1 or 0 is to be written, is located at the intersection of the excited buses. The half-select cores at which a current of  $I_s/2$  magnitude arrives at only one winding ( $x$  or  $y$ ), will not undergo magnetic reversal or switching.

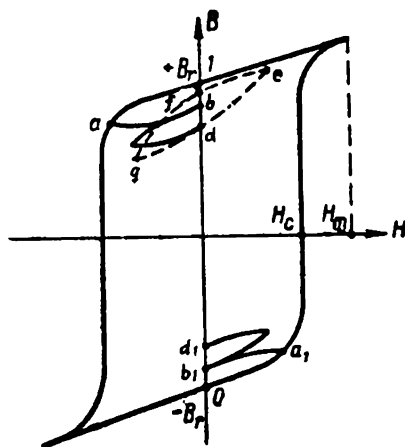


Fig.122 Mechanism by which the Data in the Half-Select Cores of a Ferrite Matrix is Erased

The reading scheme is similar to the writing scheme, except that currents of opposite polarity are fed to the  $x$  and  $y$  wires. If the code 1 was recorded in the core from which the number is being read then, under the action of the read pulses, its magnetic polarity is reversed, and a useful signal emf is induced in the common output winding. If, however, the number 0 was written in the selected core, a noise emf is instead induced in the output winding. /228

The reading scheme is a destructive one, i.e., the information stored in the matrix is destroyed; the cores of the matrix are reset to zero position. Reading must therefore be followed by rewriting (regeneration) if the information is to be preserved.

Under the action of the read pulse, the magnetic state of the half-select cores in which the code 1 is stored, varies along the hysteresis loop from the

point  $+B_r$  to the point  $a$ , and then, after the action of the magnetizing force has ended, to the point  $b$  (Fig.122). The positions of the points  $a$  and  $b$  is determined by the strength  $H$  of the magnetizing field and by the character of the hysteresis loop. Consequently, the remanence of such cores is decreased. Further than that, the repeated action of read pulses may gradually lead to a demagnetization of the half-select cores, i.e., may erase the information. Similarly, the repeated action of write pulses for the code 1 may disturb the information in the half-select cores where zeros are stored. The repeated action, on half-select cores where code 1 is stored, of nonswitching read and write  $I_r/2$  current pulses will lead to the formation of a closed partial hysteresis cycle (defg in Fig.122). For cores with an almost rectangular hysteresis loop, the area of these cycles is rather small and the asymptotic points of variation in induction are very close to the value of the original remanent induction.

This results in more severe requirements for the core materials to be used in such matrices. The squareness ratio of such core materials must be close to unity, and the remanence, on repeated action of nonswitching pulses, must not differ significantly from a certain value close to that of the original remanent induction. Since magnetic materials with an ideal rectangular hysteresis loop do not exist, the requirements as to the scatter of the amplitude of the write and read pulses must be rather severe. /229

If the shape of the hysteresis loop deviates from rectangular, the total noise signal induced by all the half-select cores in the common output read winding may be comparable to the useful signal obtained as a result of the reading of the code 1. This would make it impossible to identify the stored information. To increase the signal-to-noise ratio on reading one to the ratio on reading zero, the common output winding of the matrix is coupled with the cores in such a way that the noise induced in it by each pair of half-select cores will be mutually compensated (see Fig.121). This decreases the total noise signal by many times, but if the memory unit has a large capacity, the signal will still retain a considerable magnitude since different information is as a rule recorded in the half-select cores. Obviously in this case, in reading the code 1, either a positive or a negative signal will be induced in the output winding, depending on the sense in which the output winding passes through the selected core of the matrix. Since other units of the computer (for example the arithmetic unit) operate with codes in which a signal of definite polarity corresponds to a one and the absence of a signal to a zero, an amplifier-shaper circuit must be connected to the output of the matrix to react to signals of both polarities but to pass only signals of one polarity (code 1).

The emf in the output winding of a square matrix (see Fig.121) can be defined by the formula

$$e_{out} = \pm [e_1 - 2e_2 + (n - 2)e_3], \quad (50)$$

where  $e_1$  is the emf induced in the read winding on magnetic reversal or switching of the selected core;

$e_2$  is the noise emf induced on switching the selected core;

$e_3$  is the difference between the noise emf's formed on switching two cores

on the excited buses and located differently relative to the output winding of the matrix;  
 $n$  is the number of coordinate  $x$  or  $y$  buses.

Equation (50) is valid if  $n$  is even. An increase in  $n$  leads to an increase in the noise.

Circuit with shifted write current. The circuit of a matrix operating on the principle of a shift in the write current is exactly the same as that of a matrix with coincident currents. The operating principle of the selected core of the matrix can be explained on hand of timing charts (Fig.123). During <sup>/230</sup> the read period, the negative current pulses  $I_x$  and  $I_y$ , each of amplitude  $I_s/2$ , are combined, forming a negative current pulse of amplitude  $I_s$ .

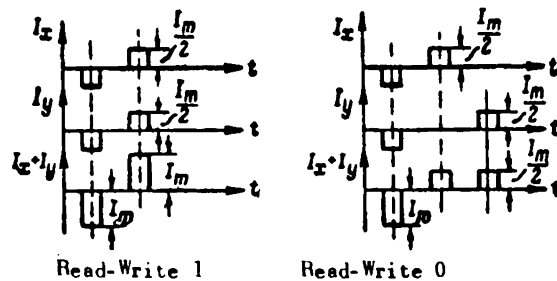


Fig.123 Timing Charts for Operation of a Matrix Based on the Principle of Write-Current Shift

In recording the code 1, the positive current pulses  $I_x$  and  $I_y$  are combined, forming a switching current pulse of amplitude  $I_s$ . In recording the code 0, the positive current pulses  $I_x$  and  $I_y$  are shifted relatively to each other, and therefore the selected core is not switched but remains in zero position after the reading.

Dynamic bias circuit. In the ferrite matrices considered above (see Fig.121), the ratio of the amplitude  $I_s$  of the switching current to the amplitude  $I_s/2$  of a (half-select) current which will not appreciably change the magnetic state of the core is 2. Let us denote this ratio by  $\alpha$ . Since actual ferrite cores have characteristics different from the idealized rectangular loop, it follows that we can judge from the value of the factor  $\alpha$  the probability of an undesirable change of state of the cores under the action of current pulses of amplitude  $I_s/\alpha$ . To decrease the probability of erasure by half-select  $I_s/\alpha$  currents, and also to decrease the noise,  $\alpha$  may be increased.

One of the simplest methods of increasing  $\alpha$  is by using the principle of dynamic bias. A matrix with dynamic bias (Fig.124a) has a conductor passing in the same direction through all the cores. Through this conductor flows the dynamic bias current  $I_b$ , of amplitude  $I_s/3$ , which is always of opposite direction to the currents  $I_x$  and  $I_y$ . The amplitude of the currents  $I_x$  and  $I_y$  is

taken as  $\frac{2}{3} I_s$ .

It is clear from the timing charts (Fig.124b) which illustrate the operating principle of a selected core of the matrix that, on reading or writing the code 1, a total current of amplitude  $I_s$  becomes effective and, on writing the

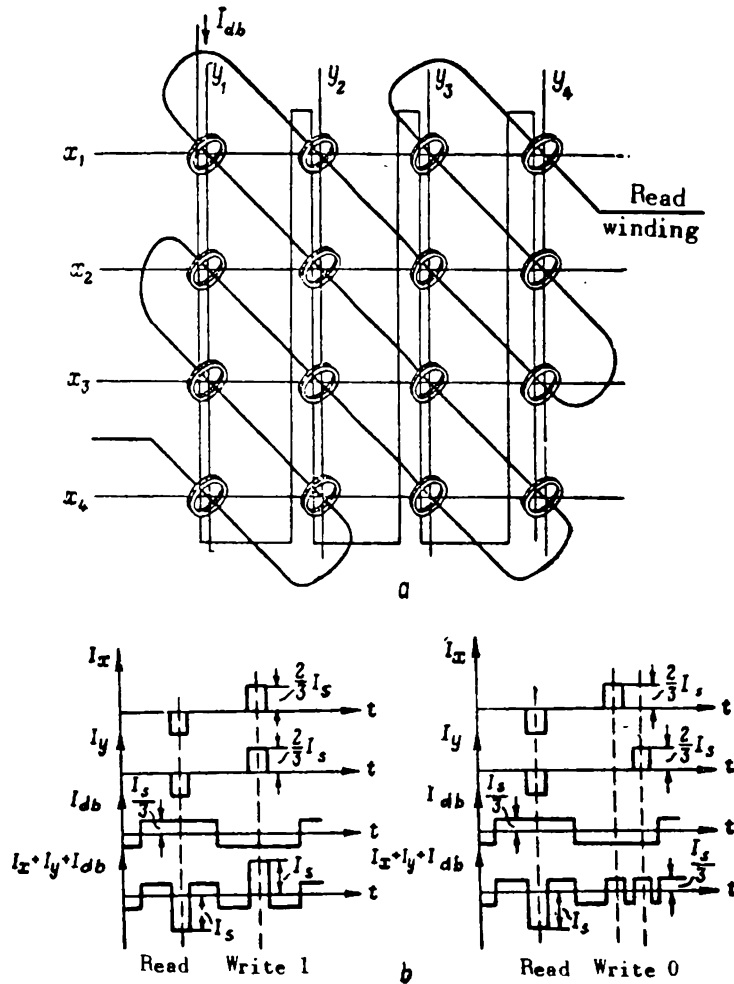


Fig.124 Circuit of Matrix with Dynamic Bias (a) and Timing Charts of its Operation (b)

code 0, a total current of amplitude  $I_s/3$ . It must be borne in mind that, in writing the code 0, the currents  $I_x$  and  $I_y$  are shifted relative to each other. For the half-select cores lying on the excited bus, the value of the parasitic current will not exceed  $I_s/3$ , since this current equals the difference of the currents  $I_x$  (or  $I_y$ ) and  $I_{db}$ . Finally, all the other cores outside of the excited coordinate buses are subjected to a bias current of amplitude  $I_s/3$ . /231 Consequently, for all cores of the matrix,  $\alpha = 3$ .

Let us consider the design principle of the matrix circuit of a MWMU for storing multidigit numbers, operating on the principle of coincident currents,

which circuit is now widely used.

Matrix-type MWMU for multidigit numbers. In order to build a matrix-type MWMU, the matrices are combined to form a matrix cube (Fig.125). Since each matrix is used to store the digits of one particular place, the number of matrices equals the number of places in the numbers to be stored. The codes of the numbers are simultaneously recorded and selected from all the matrices. For

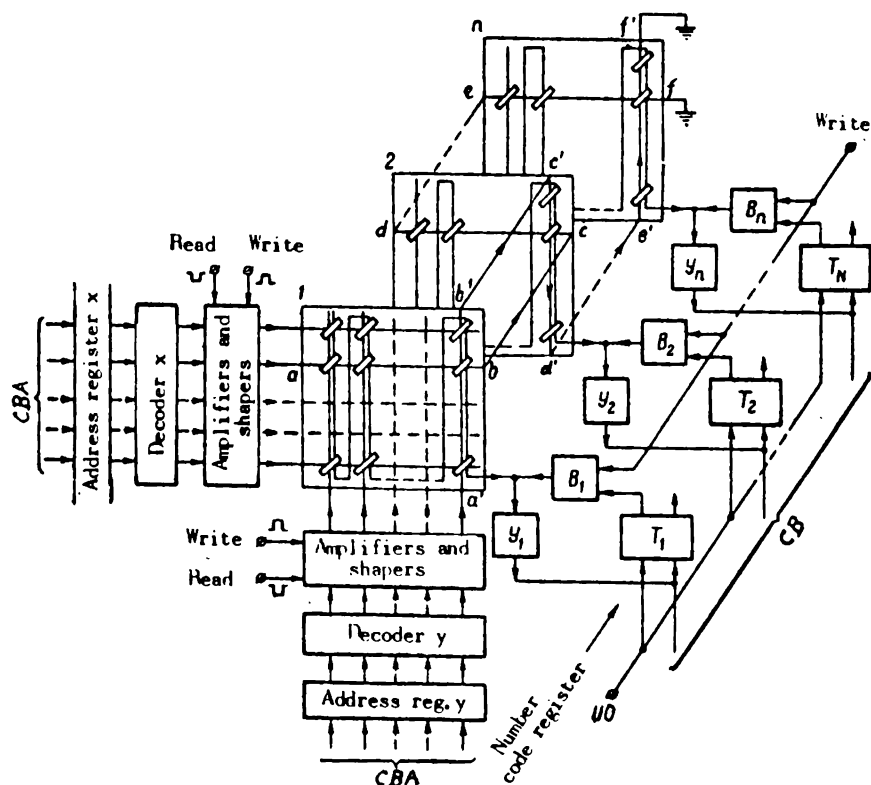


Fig.125 Matrix-Type MWMU for Multidigit Numbers

this purpose, the windings of all the cores lying on the same horizontal level in different matrices of the cube are connected in series, as shown on the diagram by the broken line  $abcdef$ . The windings of the cores lying on the same vertical plane (the broken line  $a'b'c'd'e'f''$ ) are similarly connected. Each of the series-connected circuits on the horizontal and on the vertical is fed from a separate current source, an amplifier in the amplifier module. The common read winding of each matrix is connected to the read amplifiers  $A_1, A_2, \dots, A_n$  which amplify signals of one polarity but are insensitive to signals of the opposite polarity, and to the write gates  $B_1, B_2, \dots, B_n$ . It must be borne in mind that, in the matrices of this memory unit, the common read winding is not coupled with the magnetic cores in the manner shown in Fig.121 but rather passes through all the cores of the matrix in only one direction. For this reason, a signal of only one polarity corresponds to the code 1 in the read winding.

The code of a number is selected from the memory cube by the following method: The address code of the number is entered in the address registers  $x$  and  $y$  and is then fed to the decoders. On one of the output buses of each decoder (their number is equal to the number of address buses on the coordinates  $x$  and  $y$ , respectively) there appears a signal which is amplified and then routed to the address buses of the matrices. The cores which are located at the intersection of the excited buses and in which the code 1 has been written, are switched. In this case, in the read winding a pulse is obtained which is amplified and recorded by the flip-flop of the number code register. The cores that are in zero code position remain in that position. Thus, the number code selected from the cube is stored in the register and transmitted from it to the arithmetic unit.

During the reading, the information in the memory cell of the cube is erased. To regenerate it, write pulses of polarity opposite to that of the read pulses are fed to the address buses that had previously been under the action of the read pulses. The write pulses are fed after the read pulses on reception of the signal "write" which arrives in the amplifier and reshaper module a short time after the signal "read". In order to return into code position 1 only those cores which before the reading had a 1, an inhibit pulse is fed to the common read winding of the matrix where the code 0 is to be stored.

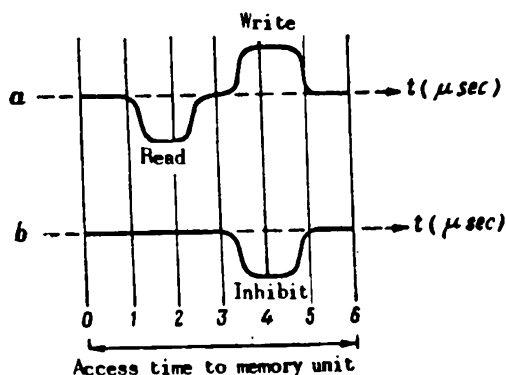


Fig.126 Timing Chart for Reading and Regeneration of the Code of a Number in a Memory Cube of a Matrix-Type MWMU

The inhibit pulses are taken from the gates  $B_1, B_2, \dots, B_n$  at the instant the signal "write" is given. The gates control the flip-flops of the number code register. They are unblocked only for those flip-flops that are in the position corresponding to the code 0 in the given place of the number. The inhibit pulses compensate the action of the write pulses arriving on the address buses, and thereby prevent the cores from passing into the code position 1.

If the information is read without regenerating, no write pulses (following the read pulses) are fed to the address buses.

To write a new number in the free cells of the memory cube, the code of the number is fed over the code buses CB and recorded in the register. The position

(address) where the number is to be recorded is selected in the same way as in reading. In this case, only the signal "write" is fed to the amplifier and reshaper module.

Figure 126 is a timing chart of the reading and regeneration of a number code in the memory cube. The pulses a are taken from the module of amplifiers and reshapers and are routed into the cube along one of the address buses. A pulse of negative polarity is used for reading, and a pulse of positive polarity for writing (regenerating) the code. The inhibit pulse b is of negative 23 polarity and compensates the action of the positive write pulse.

There are several working memory units of the matrix type in existence, which function on the above principle. For example, the Massachusetts Institute of Technology has developed a MWMU for storing 65536 numbers, each of 19 digits. The plates use ferrite cores of the S-1 brand. The cores have an outer diameter of 2 mm, an inside diameter of 1.27 mm, and a thickness of 0.6 mm. Magnetic decoders with tape cores, having a rectangular hysteresis loop, are used to control this MWMU. The access time is 7  $\mu$ sec. The control circuit contains 425 vacuum-tube duo-triodes and 625 transistors.

A matrix-type MWMU with a vacuum-tube control circuit is also used in the Soviet "Ural-2" computer.

Matrix-type MWMU have recently been developed with only magnetic and semiconductor elements used for the control circuit.

This memory unit belongs to the category of MWMU with writing and reading by half-select currents. Below, we will give the main shortcomings of such a unit.

1. Cores with a high squareness ratio of the hysteresis loop must be used for the ferrite cube. Moreover, only a slight scatter of the core parameters is permissible. This makes it necessary to select the cores individually.
2. The amplitude variation of the write, read, and inhibit current pulses and the accuracy of their synchronization must meet severe requirements. For example, the MWMU of this type which is used in the "Ural-2", with a half-current  $0.5 I_s$  of an amplitude equal to 0.5 amp, permits a deviation of not over 5%. The duration of the write, read, and inhibit current pulses is of the order of 0.1  $\mu$ sec.
3. The signals received at the output of the ferrite matrices in reading 1 and 0 do not differ in polarity (as is the case, for example, in an MU using potentiometers and which is better from the point of view of the reliability of signal identification) but in magnitude: To the code 1 corresponds the useful signal, and to the code 0 the noise signal. This makes it necessary to use 23 additional circuits and systems to increase the signal-to-noise ratio, which involves further complication of the MWMU.
4. To compensate the noise from the half-select cores of the ferrite matrices, a highly complex (diagonal) wiring of the cores is necessary.



The advantages of a matrix-type MWMU with half-current writing and reading are as follows:

a) Relatively high speed, since random access to the required memory cell in writing and reading is possible (the search time for the cell is zero), and also short access time to the MU;

b) long life of the ferrite cube;

c) possibility for large capacity of the MU (up to several thousand numbers).

The erasing of the information on reading is characteristic of a matrix-type MWMU, as it is of all MU with ferrite cores having a rectangular hysteresis loop. Regeneration circuits must therefore be added to regenerate the recorded information, which naturally complicates the MWMU.

### Section 35. Magnetic Working Memory Units with Half-Select Current Writing and Full-Current Reading

In a matrix-type MWMU (Fig.125), the principle of coincident currents of amplitude  $0.5 I_s$  is used for both writing and reading; its use for reading involves the production of noise from the half-select cores. In MWMU with half-current writing and full-current reading, the principle of coincident currents is used only in writing and regeneration, while reading is accomplished by feeding a full switching current pulse of amplitude  $I_s$ . Since there are no half-select cores during the read period, a MWMU of this type gives a higher signal-to-noise ratio.

Figure 127 is a block diagram of such a MWMU with half-current writing and full-current reading. The basic element of this MWMU is a flat matrix of ferrite cores. The cores of the matrix, arranged in one horizontal row, form a memory cell for storing one binary number. The cores in one vertical column serve to store the same digits of all the stored numbers. The horizontal buses, each passing through the cores of a single memory cell, are called address buses, while the vertical buses are called digit buses. The cores in one column have a common output winding. /236

To select the required cell in writing or reading, the address of the cell is entered in the address register. The code of the address is decoded by the address decoder, producing a signal at one of its outputs. After amplification and shaping, this signal is routed to the address bus.

To read or write a number, a pair of current pulses is fed to the selected address bus (Fig.128a). The first pulse, of amplitude  $I_s$ , sets all the cores on the selected bus to code position 0 - the number is read out. Under the action of the second pulse, of amplitude  $\frac{1}{2} I_s$ , the read information is regenerated by the coincidence of this pulse with similar pulses arriving at the same

digit buses where the code 1 had been recorded previously. When a new number is written, the second pulse thus permits recording the code 1 in the appropriate cores.

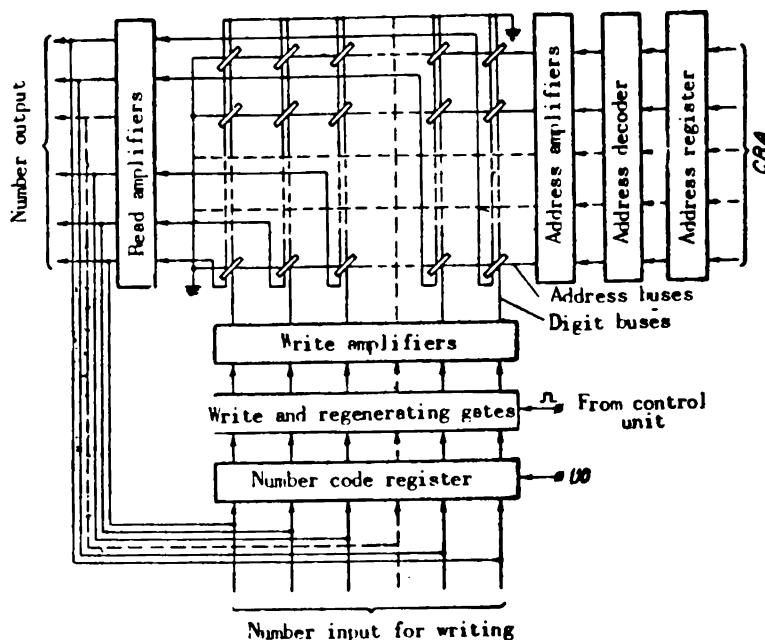


Fig.127 Block Diagram of MWMU with Half-Current Writing and Full-Current Reading

A number read from the matrix in parallel code arrives at the read amplifier, where the code pulses of the number are amplified and fed to the output of the unit. The number being read is also fed for rewriting to the number-<sup>237</sup> code register whose flip-flops have first been reset by a "0" pulse to the code position 0. When a control signal is fed from the control unit to the read and regenerating gate, the code pulses of the number are passed from the register to the read amplifier module and then to the digit bus. The amplitude of the code pulses in the digit buses is  $\frac{1}{2} I$ , (such pulses are capable only of producing half-excited cores), and they coincide in time with the second pulse arriving on the selected address bus (the pulse  $\frac{1}{2} I$ , in Fig.128a). If the corresponding place of the number contains the code 0, then no signal appears on the digit bus. The new number is similarly recorded. The only difference is that it is not the read-out number that is recorded in the register, but the new number arriving on the code buses.

Here, as in the matrix of Fig.121, the remanence in the cores subjected to the action of the half-select currents can be diminished. In fact, on the

cores located along a single digit bus, which serves to store the same digit position of all the stored numbers, pulses of half-current  $\frac{1}{2} I_s$ , taken from

the corresponding write amplifiers, may stay effective an indefinite number of times, with writing or regeneration of ones being accomplished according to other addresses in the cores of this place. At monopolar half-excitation, when pulses of half-current of one polarity are fed to the digit buses, this may decrease the remanence in the cores that had been in the code position 0.

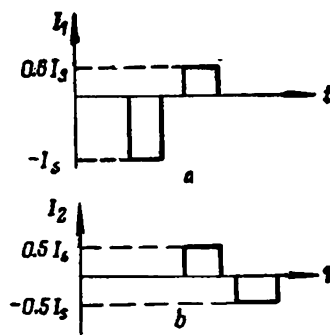


Fig.1.28 Current Pulses in Address and Digit Buses of MWMU with Half-Current Writing and Full-Current Reading  
a - Current pulses in address buses; b - Current pulses in digit buses

To avoid erasing the zero, bipolar currents of the same amplitude (see Fig.1.28b) are fed to the digit buses in writing or regenerating the code 1, instead of monopolar pulses (see Fig.1.28b). In this case, both the cores storing the code 0 and the cores storing the code 1 are subjected to the demagnetizing action of the half-select current. In both cases, however, this leads to the formation of a closed partial hysteresis cycle similar to that shown by the letters defg in Fig.1.22.

Active resistances are connected in the address and digit buses to stabilize the currents there. Since the number of switched cores on each address bus may be different, the resistors connected to the buses are of a type that 238 will keep the current constant under varying load.

An experimental study of the matrix has shown the possibility of obtaining nondestructive storage in the matrix by using square current pulses of 2 - 5  $\mu$ sec duration and 60 - 120 ma amplitude (for the half-currents). The optimum signal-to-noise ratio is obtained with current pulses of about 100 ma and 5  $\mu$ sec in duration. Reliable operation is possible even at great instability of the amplitude of the currents ( $\pm 30\%$ ) and of their duration ( $\pm 50\%$ ).

In a MWMU with half-current writing and full-current reading, the signal-to-noise ratio is higher than in a matrix type MWMU, since there are no half-select cores. Further than that, the code 1 in the output winding is represented by a current pulse of one polarity, which simplifies the reading amplifier system.

Magnetic working memories with half-current writing and full-current reading, built according to the circuit shown in Fig.127, have only a small capacity (several tens of numbers), which is their basic shortcoming. With increasing capacity, the amount of hardware required increases sharply. /23

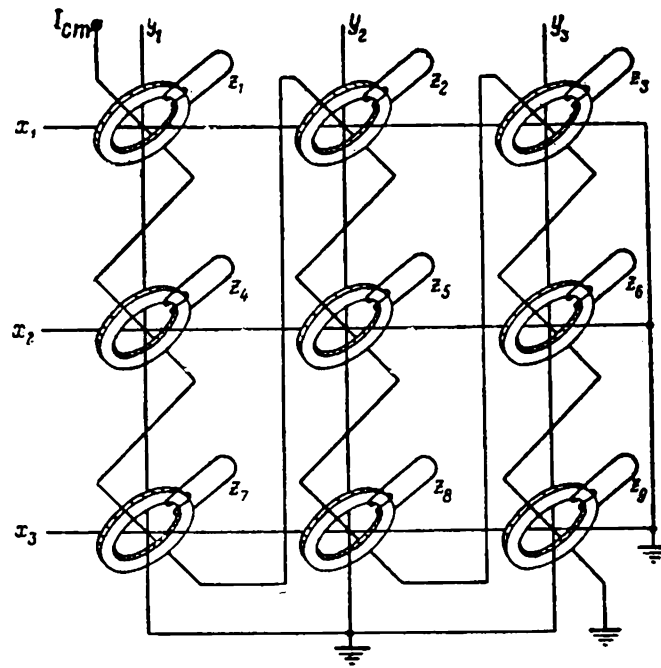


Fig.129 Ferrite Matrix of Coordinate Transformers of Type Z MWMU

Working memory units using the principle of half-current coincidence for writing and regenerating numbers and the full current for reading, include the so-called type Z MWMU. In design principle, however, the type Z MWMU differs considerably from the system we have just considered (Fig.127); this makes this unit an independent type of working memory, particularly in view of the various possible modifications. A type Z MWMU is used in the Soviet BESM-2 computers and in others.

The main building block of a type Z MWMU is a ferrite cube enclosing a matrix of coordinate transformers and number wires or lines.

The ferrite matrix, composed of coordinate transformers (toroids) is shown in Fig.129. Through the cores of these coordinate transformers pass coordinates, or address buses  $x$  and  $y$ , forming the windings  $w_x$  and  $w_y$ . Each coordinate transformer has a closed loop winding  $w_z$ , which is denoted by  $z$  (Fig.129). The winding  $w_z$  passes through a series of ferrite code cores used to store the codes of binary digits. The windings  $w_z$  with the code cores constitute the memory cell for one  $n$ -digit number. This cell is termed the number line.

The addresses of the numbers recorded in the number lines are the addresses of the coordinate toroids of which they are composed. A common superposed magnetization winding passes through all these coordinate toroids. The

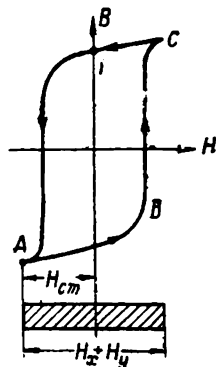


Fig.130 Position of Operating Point A on the Hysteresis Loop of Coordinate Transformers

direct current  $I_{cm}$  passing through it produces the magnetizing ampere-turns  $aw_{cm}$ , thereby determining the operating point A of the toroid (Fig.130). The

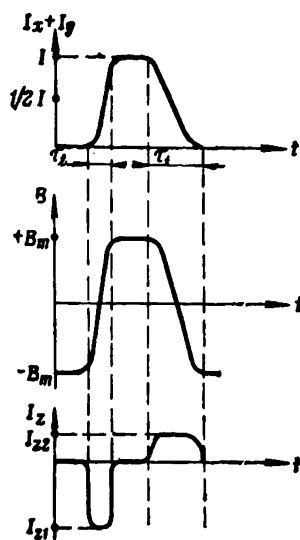


Fig.131 Timing Chart of Formation of Currents  $I_{x1}$  and  $I_{x2}$

principle of coincidence of the currents  $I_x$  and  $I_y$  flowing through the address buses x and y is used to switch the coordinate toroids. After reversal of magnetization, the toroid is returned to its original state of magnetic saturation

by the aid of constant superposed field magnetization. For selecting the required coordinate toroid, such as in the matrix-type MWMU, address registers and decoders for the coordinates  $x$  and  $y$  are used, together with amplifier-shaper systems at the output of the decoders.

When the core of a coordinate transformer is switched under the action of the total magnetic field formed by the currents  $I_x$  and  $I_y$  (switching along the curve ABC of the hysteresis loop, Fig.130), an emf producing the current  $I_{z1}$  is induced in its winding  $w_z$  (Fig.131). The amplitude of the current  $I_{z1}$  is such that the resultant magnetic field is able to switch the ferrite cores of the 240 number line. The current  $I_{z1}$  is used in the MWMU as a read current.

After the action of the current pulses  $I_x$  and  $I_y$  has ceased, the core of the selected coordinate transformer is returned to its original state (point A of the hysteresis loop) along the curves CDA under the action of the magnetic field of the current  $I_{c1}$ . In this case, an emf induced in the winding  $w_z$  causes the appearance of the current  $I_{z2}$  (Fig.131), of opposite sense to the current  $I_{z1}$ . The amplitude of the current  $I_{z2}$  is slightly more than half that of the current  $I_{z1}$ . Consequently, the magnetic field produced by the current  $I_{z2}$  in the cores of the number line are not able to switch them. The current  $I_{z2}$  is used in the MWMU as one of the half-select currents in writing and regenerating the numbers.

This ratio of amplitudes of the currents  $I_{z1}$  and  $I_{z2}$  is attained in the following manner.

Since the variation in the magnetic induction of the ferrite core depends on the time rate-of-change of the switching current, i.e.,

$$B = B[I(t)],$$

the following relation is valid:

$$e_z = -K \frac{dB[I(t)]}{dt} = -K \frac{dB[I(t)]}{dI(t)} \cdot \frac{dI(t)}{dt},$$

where  $e_z$  is the emf induced in the winding  $w_z$  of the coordinate transformer which is subject to the action of the total current  $I_x + I_y$ ;

$K$  is a proportionality factor;

$\frac{dB[I(t)]}{dI(t)}$  is a derivative characterizing the material of the core;

$\frac{dI(t)}{dt}$  is a derivative characterizing the slope of the leading edges of the switching current pulse.

Consequently, the emf  $e_z$  depends not only on the magnitude of the total current pulse but also on the slope of its leading edges.

In a type Z MWMU the steepness of the leading edge of the current pulses

$I_x$  and  $I_y$ , and consequently also that of the total current, is considerably greater than the steepness of the trailing edge (Fig.131). The emf  $e_{z1}$ , induced in the winding  $w_z$  during the rise of the total current, is therefore considerably greater than the emf  $e_{z2}$  induced in the same winding during the fall of the current. Since the currents  $I_{z1}$  and  $I_{z2}$  are, respectively, proportional to the emf  $e_{z1}$  and  $e_{z2}$ , it naturally follows that  $I_{z1} > I_{z2}$ . /241

Since the principle of coincident currents  $I_x$  and  $I_y$  is used in selecting the coordinate transformers, there will be both selected and half-selected transformers coupled to the excited bus x or y. Owing to the nonsquareness of the hysteresis loop of the core of a half-select transformer, a noise emf will arise in its winding  $w_z$ , which could have a bad effect on the operating reliability of the number line during writing and reading.

To compensate the noise currents produced in the half-select toroids of the matrix, each working coordinate toroid is provided with an additional compensating toroid. The connection of the windings of the working and compensating toroids is the same as that used to compensate the noise due to deviations in the shape of the hysteresis loop of the magnetic material from the rectangular form, in the multistage magnetic decoder (see Fig.92).

It is well known that the resistance of a core to current flowing through its winding will vary, depending on whether the magnetic polarity of the core is reversed or not. This is particularly marked in magnetic materials with a rectangular hysteresis loop: the resistance of a magnetic core in reading the code 1 may be several times as great as in reading the code 0, when the magnetic state of the core is not changed.

This brings up the problem of stabilizing the current arriving through the winding  $w_z$  from the coordinate toroids to the number lines. This problem can be solved readily enough, in principle, by connecting a stabilizing resistor in series with the memory cores of the number line. With a large number of digits in the numbers, however, such a solution is disadvantageous from the energetic viewpoint. It is better to use stabilizing cores instead of a stabilizing resistor. In this case, the number line (Fig.132) has two types of magnetic cores: working cores to store the digits of the storage numbers, and stabilizing cores for stabilizing the load across the coordinate toroid. To each working core there corresponds its own stabilizing core. The winding  $w_z$  passes through these cores in opposite senses. Thus, when a current pulse  $I_{z1}$  appears in the winding, and switches these cores, the working and stabilizing cores are magnetized in different senses: the working core (WC) is switched to the code position 0 ( $-B_r$ ) and the stabilizing core (SC) to the code position 1 (corresponding to  $+B_r$ ).

To hold the load on the coordinate toroid constant, each pair of cores of the number line must be in the same magnetic state before reading the data. In that case, regardless of the code recorded in the line, one of the cores of each place of the number, on reading, is switched to the opposite state of saturation, while the other remains in the same state or is switched over a partial /242 hysteresis cycle. After passage of the read pulse  $I_{z1}$  through the winding  $w_z$ , all the working cores are set to the code position 0, and all the stabilizing

cores to the code position 1. If one is then recorded in some place, the working core of the corresponding pair is set to the code position 1, while the stabilizing core must, as before, remain in the same position. But if zero is recorded, then the working core remains in its previous position while the

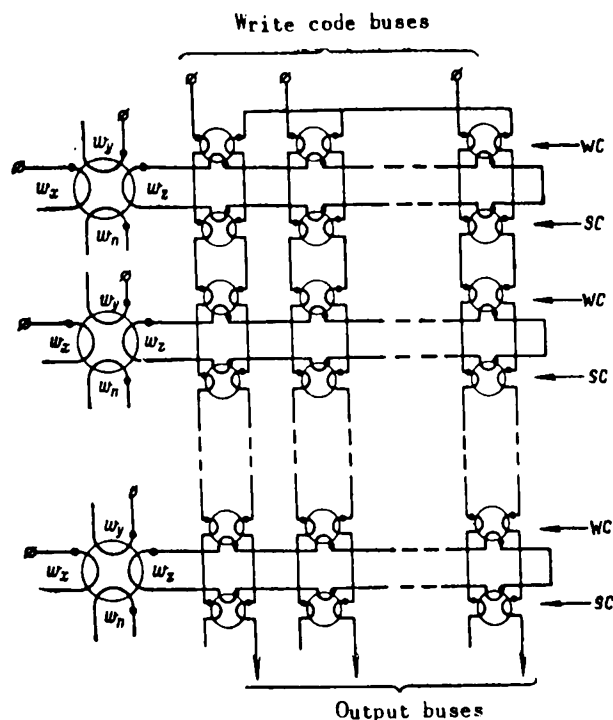


Fig.132 Number Lines in a Type Z MWMU

stabilizing core is switched from position 1 to position 0. Only in this case will the magnetic cores of each pair after recording be in the same states and, consequently, the load on the coordinate toroid from the number line will remain constant.

Each core of the number line has two windings besides the winding  $w_x$ , the write winding, into which pulses are fed when the codes are recorded in the place involved, and the output (read) winding, in which an emf corresponding to the read-out code is induced. Both of them are common to the cores of all the number lines of the memory unit designed to store the same place of the 243 numbers. Consequently, the number of code write buses and read buses of an MWMU is determined by the number of digits in the numbers to be stored.

When the memory unit is operating in the writing mode, code write pulses  $I_x$  are applied through the write windings when the pulse  $I_{x2}$  passes through the winding  $w_x$ . When the codes 1 and 0 are recorded, the cores of the number line are switched into the corresponding position under the action of the total magnetic field created on simultaneous passage of the pulse  $I_{x2}$  and of the code write pulses (the latter, by themselves, are not powerful enough to switch the cores).



Figure 133 gives timing charts to illustrate the operation of the memory unit in various modes. The graphs A and B show the total values of the current pulses in the winding  $w_z$  of the coordinate toroid selected and in the write windings for the working and stabilizing cores of the number line. Graph C shows the magnitude and time of application of the code pulses  $I_k$ . Graphs D and E show the current pulses in the output winding, which arise on switching

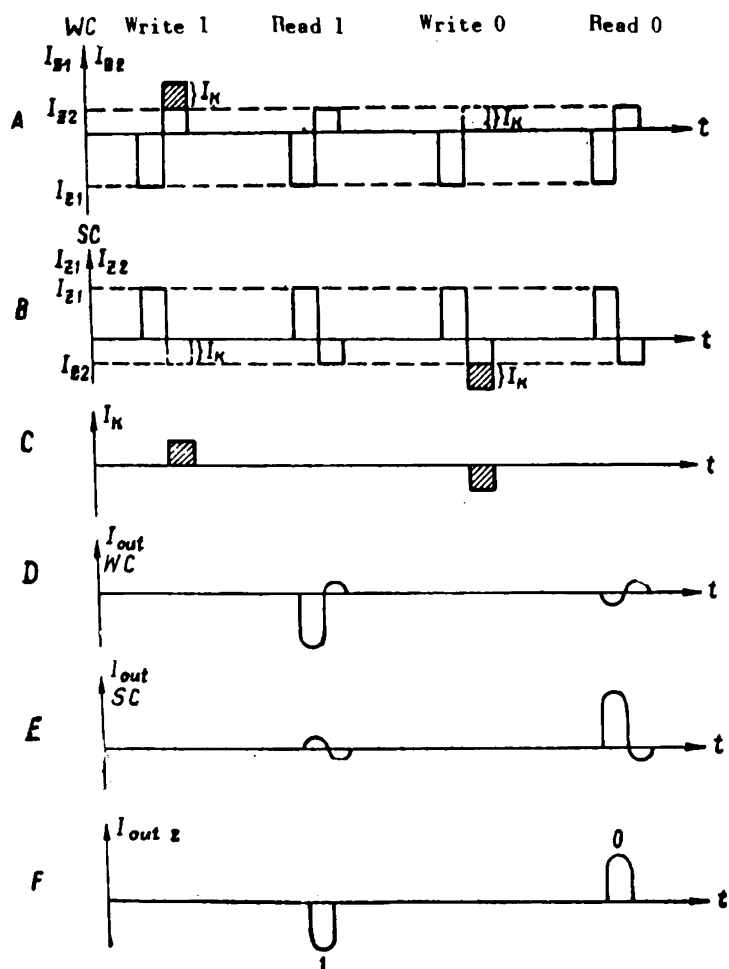


Fig.133 Timing Charts of Operation of a Type Z MWMU in Various Modes

the working and stabilizing cores at the instant of reading. Graphs F shows the total value of the pulses in the output winding.

In recording the code 1, when the pulse  $I_{z2}$  is transferred, a positive code pulse  $I_k$  is applied. Under the action of the total magnetic field due to the pulses  $I_{z2}$  and  $I_k$ , the working core is set in the code position 1. In the stabilizing core, the write current  $I_k$  is subtracted from the current  $I_{z2}$ , and therefore this core remains as before in position 1, recording in it when the

pulse  $I_{z1}$  passes through the winding  $w_z$ .

When the code 0 is recorded, a negative pulse  $I_k$  is applied to the write winding. In this case the pulses  $I_{z2}$  and  $I_k$  are subtracted in the working core, while in the stabilizing core they are added. As a result, the working core remains in code position 0, while the stabilizing core is reset from the code position 1, recorded in it on passage of the pulse  $I_{z1}$ , to the code position 0.

In the reading mode, no pulses  $I_k$  are applied. Reading is by the pulse  $I_{z1}$  arising in the winding  $w_z$  of the coordinate toroid selected. If the code 1 was written there, then on reading, the working core of the number line is reset to the code position 0, while the stabilizing core remains in the position 1. In this case, a negative current pulse corresponding to the code 1 appears in the output winding (graph F, mode "read 1"). If the code 0 was recorded, i.e., if both the cores belonging to the digit in question, were in the position 0, then, on reading, the working core remains in the position 0, while the stabilizing core is set to the position 1. In this case, a positive pulse corresponding /244 to the code 0 appears in the output winding of the digit position in question.

Thus the signals of code 1 in the output windings of a type Z MWMU do not differ from the signals of the code 0 in amplitude but rather in sign. This increases the reliability of data readout. If circuits sensitive to pulses of only one polarity are connected to the MWMU output, we obtain the code of the number in which one is characterized by a pulse and zero by its absence, which is what is required for the operation of other units of the computer (especially the arithmetic unit).

Consequently, the use of a stabilizing core in each place of a number /245 line not only ensures constant load for the coordinate transformer, but also gives signals of different polarity in reading 1 and 0. In addition, it almost completely eliminates the noise due to the deviation of the shape of the hysteresis loop from rectangular. This makes it possible for a type Z MWMU to use ferrite cores of lower grade than in a matrix-type MWMU, with half-current reading and writing.

The following are the principal advantages of a type Z MWMU.

1. Possibility of designing a high-capacity working memory (up to several thousand numbers).

2. Practical absence of noise in the input buses. It is due to the fact that, in reading, the signal 1 has a polarity different from that of the signal 0. Moreover, in reading a number, the magnitude of the code signals in the output buses is determined by the change in the magnetic states of the ferrite cores of only the number line that is addressed.

3. Less severe requirements as to the parameters of the cores for the number lines.

4. Higher speed than a matrix-type MWMU. This increased speed is due to the fact that the amplitude of the read current  $I_{z1}$  may be very great. This,

in turn, makes it possible to force the switching of a core of the number line, which, besides increasing the speed of the unit, also helps to increase the code signals in the output buses.

A type Z MWMU has the following shortcomings:

a) A large number of cores is required. Thus, the MWMU ferrite array of the BESM-2 computer, with a capacity of 2048 39-digit binary numbers, takes about 200,000 cores of type K-132 for building the number lines and type K-65 cores for building the coordinate matrix. A matrix-type MWMU of the same capacity would require only about 80,000 cores.

b) The power required by the address pulse shapers, giving the  $I_x$  and  $I_y$  signals, is higher owing to the losses in the half-select coordinate transformers. The power of the address pulse shapers in the MWMU of the BESM-2 is 400 watts per pulse, while only 5 - 7% of this power is fed to the selected number line. The rest is consumed in each of the coordinate transformers.

c) The coordinate transformers operate under a severe regime. The heating and the consequent decrease in loop rectangularity also increases the noise currents. The use of compensating coordinate transformers in the coordinate matrix, together with the stabilizing cores in the number lines, which improve the noise figure of the MWMU, requires about a third more power.

### Section 36. Permanent Ferrite-Core Memory Unit

The above types of magnetic working stores cannot be expediently used to store permanent quantities (constants, values of tabular functions, program instructions). In such units, actually, each core is used to store only one code (1 or 0). Moreover, during readout, the information is erased from the memory and must be regenerated, resulting in additional complication of the MU circuit and increasing the power consumption.

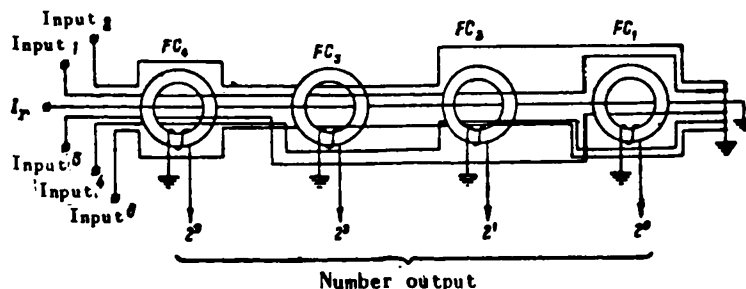


Fig.134 Number Line using Ferrite Cores for Storage of Five Four-Digit Numbers

It is desirable to have a unit for storage of constant quantities in which each core could be used to store several codes instead of only one, and from which the information could be read an unlimited number of times without

regeneration.

The number line (Fig.134) is the basis for such a unit, which is known as a permanent or long-term memory unit (PMU) using ferrite cores (Fig.134). The line shown by Fig.134 is designed to store five four-digit binary numbers (the number of input drives), but can also be used to store a larger quantity of information (up to several tens of numbers). Each core stores one digit of all the numbers stored.

Each input bus either passes through the core of a number line if the code 1 is to be stored or read in that place of the number, or bypasses it if the code 0 is to be stored or read. Each core of the line has its own output winding. A common read drive passes through all the cores. When a read pulse  $I_r$  is applied to this drive, the cores are reset to the code position 0.

The sequence of selecting the numbers recorded on a number line is as follows: Let it be required to read the first number (the running number of the number is the same as the running number of the input bus). This requires application of a read pulse to the first input bus. This pulse sets the 24 cores, through which the first input bus passes, into the code position 1 (until then, as a result of the reading of the last number, all the cores of the line had been in the code position 0). However, those cores that the bus bypasses remain in the zero position. During writing, signals of such polarity are induced in the output windings of the cores that they are not sensed by the following circuits.

TABLE 18

TABLE OF VALUES OF THE OUTPUT NUMBER

Digit No. of Number	$2^3$	$2^2$	$2^1$	$2^0$
1	1	1	1	0
2	0	1	0	0
3	1	0	0	1
4	1	0	1	0
5	0	1	1	0

Then, in the common read winding the pulse  $I_r$  is applied. As a result of switching the cores which previous to this had been in the position 1, code pulses appear in the output windings and are sensed by the following circuits. In this case, pulses appear in the output windings of the cores  $FC_2$ ,  $FC_3$ ,  $FC_4$  through which the first input bus passes. There is no signal at the output of core  $FC_1$  since the first bus bypasses this core. Thus, the code of the number 1110 will appear at the output. For a second reading of this number, a write pulse must again be applied to the input winding, followed by a pulse  $I_r$  to the read winding.

Table 18 gives the values of the numbers recorded on the line, in accordance with the sequence of passage of the input buses through its cores.

In reading information from the number line, noise will appear in the output windings of the cores from which the code 0 is taken. This noise can be eliminated by threading the output winding in a certain way through the cores used for storing the same digit positions of the stored numbers in all the number lines of the unit.

### Section 37. Transfluxors

A transfluxor is a ferrite core with rectangular hysteresis loop in which the magnetic flux can be closed in different ways, because the core has two or more apertures or windows.

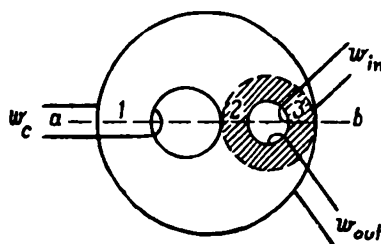


Fig.135 Transfluxor with Two Apertures

Let us consider the operating principle of such a device on the simplest example of a transfluxor with two dissimilar apertures (Fig.135). On the 248 diametral section  $ab$  of the magnetic core are the three jumpers 1, 2, and 3; the cross-sectional area of the jumpers 2 and 3 are the same, while the cross-sectional area of the jumper 1 is equal to, or greater than, the sum of the cross-sectional areas of the jumpers 2 and 3. The transfluxor has three windings: control  $w_c$ , input  $w_{in}$ , and output  $w_{out}$ .

Control current pulses are applied to the winding  $w_c$  and, according to their polarity, set the transfluxor in one of two states, cut off or open.

Let a control pulse of high amplitude and polarity be applied to the winding  $w_c$ , such that a magnetic flux of clockwise direction is established, which will magnetize the material of the jumpers to saturation (Fig.136a). Since the magnetic material of the core has a rectangular hysteresis loop, the remanent magnetic flux in the winding of the core will be about equal to the saturation flux, after the pulse has stopped. Let us then apply to the winding  $w_{in}$  an alternating current of such magnitude that the resultant magnetic field strength will be insufficient to switch the magnetic flux along the loop with the jumpers 1 and 3 (Fig.135) but will be sufficient to switch the magnetic flux along the loop with the jumpers 2 and 3. In this case, no emf will be induced in the output winding of the transfluxor, because in the first half-period,

when the magnetizing force due to the AC circulates clockwise around the small aperture, it will tend to increase the flux in the jumper 3 and decrease it in the jumper 2. In the next half-period, the magnetizing force will be directed

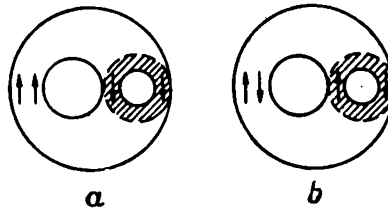


Fig.136 Direction of Magnetic Flux in a Transfluxor with Two Apertures  
a - In cutoff state; b - In open state

counterclockwise and will tend to exert the opposite action. In both cases, part of the magnetic conductor in the hatched region is already saturated, so that a further increase in the flux is impossible: During the first half-period, when the magnetizing force is directed clockwise around the small aperture, further saturation of the jumper 3 is impossible and thus an increase of flux in it is also impossible; during the second half-period, a variation of the flux in the jumper 2 is impossible. /249

Thus, after the control winding receives a current pulse sufficient to magnetize to saturation the jumpers 2 and 3, the transfluxor will be in the cutoff state. Excitation of the input winding by alternating current now does not change the magnetic flux closed along the loop 2-3-2, and no signal will be induced in the output winding. In the cutoff state of the transfluxor, the jumpers 2 and 3 are magnetized to saturation in the same direction.

A control pulse which sets the transfluxor in the cutoff state is called a blanking pulse.

To open the transfluxor, a current pulse having a polarity opposite to that of the original pulse is applied to the control winding. The magnitude of this pulse must be sufficient for magnetic reversal of the jumper closest to the winding  $w_0$ , i.e., of the jumper 2, but must be insufficient to reverse that of the jumper 3. This is explained by the fact that, at a low amplitude of the control pulse, the path of the magnetic flux through the jumper 2 is shorter than its path through the jumper 3. After passage of such a pulse, the magnetic flux in the hatched region will be directed clockwise (Fig.136b). If, now, an AC of the former magnitude is passed to the winding  $w_{in}$ , it will switch the hatched portion of the core, first in the reverse sense (counterclockwise) and then periodically, now in one sense, now in the other. In this case, an emf will be induced in the output winding, and it will become possible to transfer energy from the winding  $w_{in}$  to the winding  $w_{out}$ . This state of the transfluxor is known as unblocked or open.

Thus, in the open state of the transfluxor, when the input winding is excited by AC of this particular strength, the magnetic flux closed along the loop 2-3-2 in the hatched region will change, causing an emf to be induced in the output winding. The jumpers 2 and 3 are magnetized in opposite directions.

A control pulse that sets the transfluxor in the open state is known as a gate or unblanking pulse.

It should be noted that when AC is passed through the input winding, the alternating component of the magnetic flux in the jumper 1 is practically absent in any state of the transfluxor, i.e., the energy of the AC is not transferred into the control winding.

The magnitude of the output signal of the transfluxor is regulated by selecting the amplitude of the unblanking current pulse applied to the control winding. Assume that a transfluxor is in the blocked or cutoff state. /250

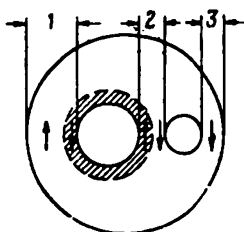


Fig.137 Mechanism of Various Degrees of Opening a Transfluxor

Let us apply an unblanking pulse to the winding  $w_c$ . Depending on its value, the jumper 2 of the core will be either completely or partially switched (Fig.137). This is due to the fact that a magnetizing force, proportional to the applied unblanking pulse and inversely proportional to the radius of the elementary layer of the annular zone around the large aperture, is created in those elementary layers.

Consequently, at a certain value of the unblanking pulse, there exists, within the magnetic loop 1-2-1 around the large aperture of the core, a circumference cutting off an inner zone (hatched) within which the magnetizing force due to the gate pulse is sufficient to reverse the sense of the magnetic flux, i.e., to switch the magnetic material of the core. In the zone outside this circle, the magnetic field strength is insufficient for magnetic reversal or switching. When alternating current is passed through the input winding, a magnetizing force is produced which is able to reverse the direction of only that part of the flux in the jumper 2 which was due to the gate pulse. The signal induced in the output winding of the transfluxor is proportional to the change of this portion of the flux, i.e., up to a certain limit it is proportional to the value of the unblanking pulse. By varying the amplitude of the unblanking pulses of the winding  $w_c$ , the signal in the output winding can be discontinuously varied.

If the amplitude of the AC in the winding  $w_{1n}$  is too great, then the magnetic field strength may be sufficient to switch not only the jumpers 2 and 3 but also the jumper 1. In this case, signals will be induced in the output winding whether or not the transfluxor was in the cutoff state; thus the transfluxor will cease to be a controlled element. This phenomenon is called spurious unblanking of a transfluxor.

To eliminate this phenomenon, it is advisable to replace the symmetric feeding of the transfluxor from the input winding by asymmetric feeding, applying stronger pulses alternately to the winding  $w_{1n}$  which will magnetize the

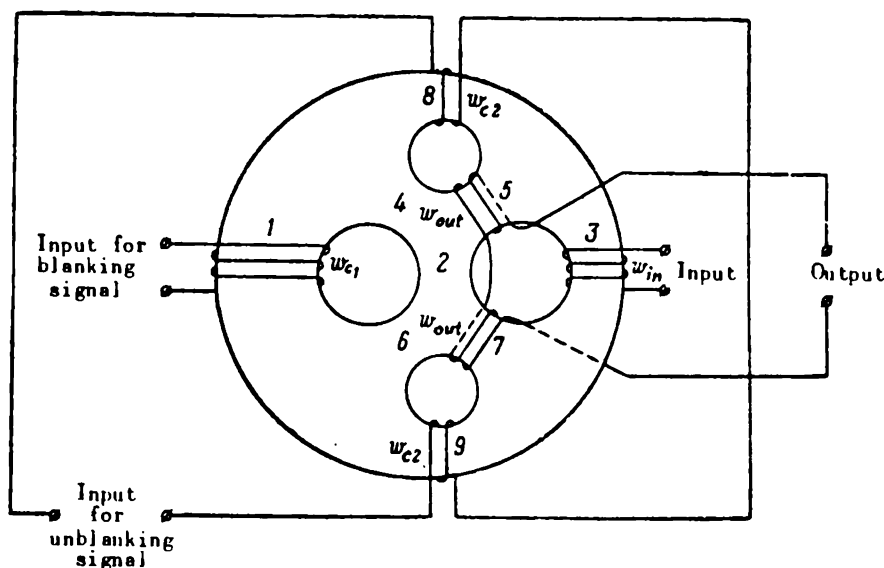


Fig.138 Transfluxor with Four Apertures

jumpers 2 and 3 in the same sense as the blanking pulse in the winding  $w_y$ , and alternately small pulses of opposite polarity, which will reverse the sense of the magnetic flux in the same jumpers. Accordingly, asymmetric feed pulses are divided into drive pulses (larger) and opposing or bucking pulses (small).

The drive pulses, no matter how great they may be, cannot cause spurious opening of the transfluxor, since no further increase of the magnetic flux is possible in the jumpers 2 and 3 if the transfluxor had been in the cutoff state until then, or in the jumper 3 if the transfluxor had been in the open state, because of full saturation. However, the magnitude of the bucking pulses must be such that the sense of the magnetic flux is reversed only in the magnetic loop 2-3-2. [25]

One of the advantages of the transfluxor with two apertures is the short time required for switching it to the cutoff or to the open state (about 2  $\mu$ sec). The repetition rate of the drive and bucking pulses or AC sinusoids in the output winding may be 1 Mc or higher without appreciable heating of the core.



One of the major shortcomings of such transfluxors is the need for limiting the amplitude of the currents in the control and input windings by an upper and lower limen.

This disadvantage can be largely compensated by using a four-aperture transfluxor (Fig.138). To set this transfluxor in the cutoff state, a blanking pulse is applied to the winding  $w_{11}$  on the jumper 1. This pulse must be sufficient to saturate the parallel magnetic loops 1-4-2-6-1 and 1-8-3-9-1. Fig- 252 ure 139 shows the sense of the magnetic fluxes in the jumpers corresponding to the cutoff state of the transfluxor. If a signal capable of reversing the flux

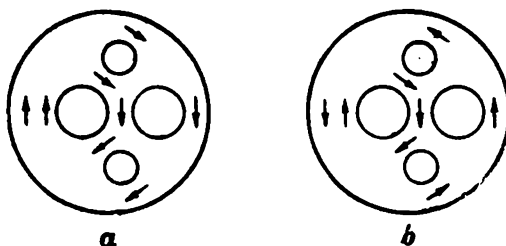


Fig.139 Sense of Magnetic Flux in a Four-Aperture Transfluxor

only along the circuit 3-5-2-7-3 is then fed to the winding  $w_{12}$ , there will be no signal at the output, since the jumpers 3 and 2 are magnetized to saturation in the same direction.

To switch a transfluxor in the open state, an unblanking pulse is fed to the winding  $w_{82}$  of the jumpers 8 and 9. This pulse reverses the sense of the flux along the loop 1-8-3-9-1. Figure 139b shows the sense of the magnetic fluxes in the jumpers when the transfluxor is in the open state.

The use of a transfluxor with four apertures makes less severe requirements on the amplitude of the control pulses, which here has only a lower limit. Consequently, the control region for such a transfluxor is practically unlimited, and it therefore operates stably over a rather wide range of ambient temperatures. This is the major advantage over two-aperture transfluxors.

Transfluxors naturally find wide use in digital computer engineering, since they have a number of favorable properties: a high-power factor, a short operating time, a negligible degree of coupling between input and output circuits. Circuits using transfluxors are highly economical.

Transfluxors can be used in a wide variety of devices as bistable elements: in registers, logic circuits, decoders, working memories, etc.

Figure 140 shows a four-aperture transfluxor for realizing the logic operation OR. In contrast to the transfluxor of Fig.138, this modification has two control windings instead of one, which both can take unblanking pulses. These pulses arrive at two pairs of terminals A and B. The transfluxor is set in

the open state if an unblanking pulse is applied either to the terminal A, or to the terminal B, or simultaneously to both pairs of terminals. In this state of the transfluxor, the passage of AC through the input winding causes an emf to be induced in the output winding until the next blanking pulse arrives. 253

The same transfluxor can also perform the operation AND if the following restriction is imposed on the magnitude of the current pulses  $I_A$  and  $I_B$  applied to the terminals A and B:

$$\frac{1}{2} |I_{c\ 3-8-1-9}| < |I_A| < |I_{c\ 3-8-1-9}|,$$

$$\frac{1}{2} |I_{c\ 3-8-1-9}| < |I_B| < |I_{c\ 3-8-1-9}|,$$

where  $I_{c\ 3-8-1-9}$  is the current strength in the control winding, creating along the loop 3-8-1-9 a magnetic field strength equal to the coercive force of the material.

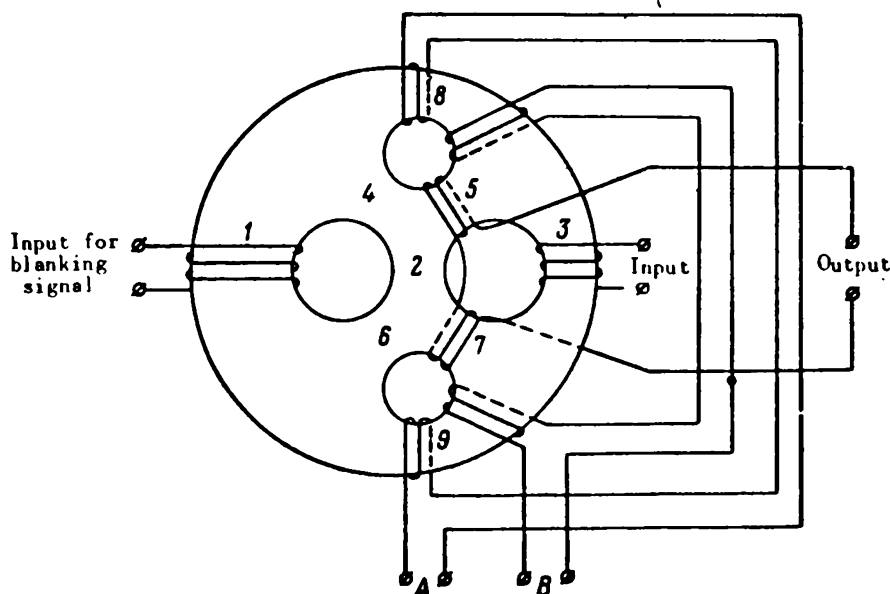


Fig.140 Transfluxor Circuit Realizing the Logic Operation OR

In this case, the transfluxor is set in the open state only when the signals across the terminals A and B are the same.

The use of transfluxors in the design of a matrix-type working memory is of great interest. Figure 141 shows an element of such a unit, a matrix consisting of transfluxors with two apertures. Here, as in the matrix of conventional cores, the current-coincidence principle is used. The only difference is that the circuits for the write pulses and the read pulses differ. In each transfluxor of the matrix, two write windings pass through the large aperture

and two read windings through the small one.

When a one or a zero is recorded, the write pulses are simultaneously /254 fed to the vertical and horizontal buses; in setting the selected transfluxor in a definite state, the full action of these two pulses is necessary. If as a result of this action, the transfluxor is set in the cutoff state, it is considered to store a zero whereas, if it is set in the open state, this means that it is storing a one. Consequently, the state of the selected core is determined by the polarity of the write pulses.

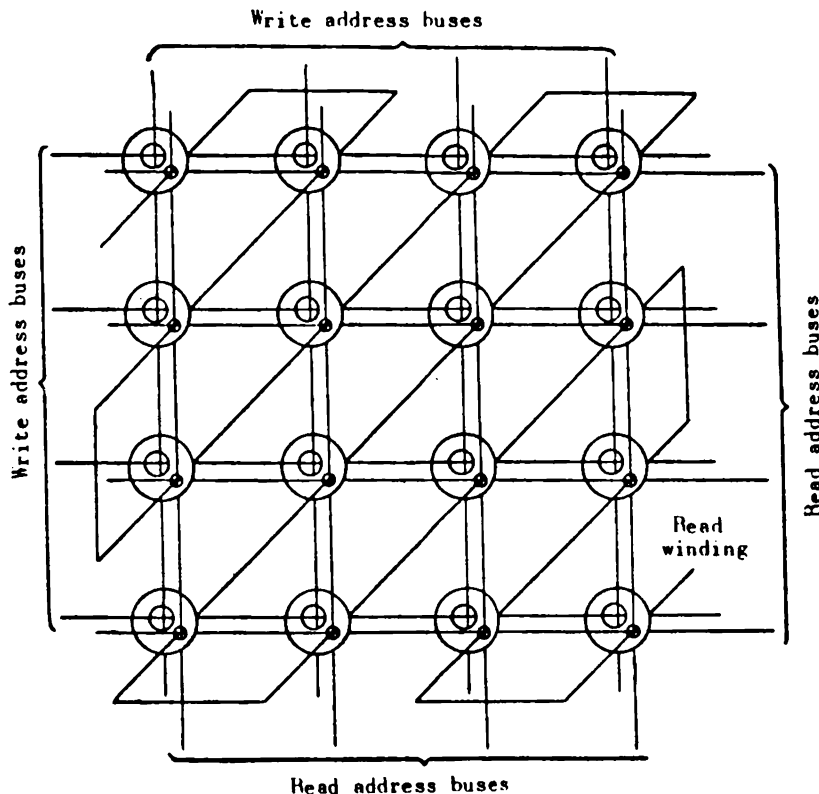


Fig.141 Matrix of Working Memory with Two-Aperture Transfluxors

Reading, which is also based on the current-coincidence principle, is performed by two bipolar pulses; one of them is applied to the vertical bus and the other to the horizontal bus. If the code 1 was written in the selected core, then the combined action of the read pulses switches the jumpers 2 and 3 (see Fig.135) and an emf is induced in the output winding of the transfluxor /255 connected to the common read winding of the matrix. If the code 0 is stored in the selected core, then during reading the jumpers 2 and 3 are not switched and there will be no signal at the output. Essentially, the read pulses are the first parts of the bipolar pulses applied to the read address buses; their second parts serve to regenerate the states of the jumpers 2 and 3 that had been reversed by the first parts of these pulses.

The information recorded in the matrix is not destroyed during reading. This is due to the fact that the sense of the magnetic flux in the jumper 1 (cf. Fig.135), determining the state of the transfluxor, is not reversed by the read pulses.

The possibility of repeated reading of information without regeneration, which permits simplification and speed-up of operation of a working memory unit with transfluxors constitutes a notable advantage of such a unit over a MWMU with conventional ferrite cores. Another no less important advantage is the possibility of writing into and reading from two different addresses. This decreases the access time and simplifies the logic of several types of computers.

The development of the transfluxor has greatly widened the field of application and increased the diversity of circuits based on magnetic elements.

### Section 38. Ferroelectric Memory Units

The term ferroelectric or piezoelectric is applied to crystalline nonconductors in which the dependence of the polarization  $P$  on the strength  $E$  of the external electric field is of the same character as the magnetization curve for ferromagnetic materials. Ferroelectrics are considered the electric analogs of ferromagnetics.

The discovery in 1944 of barium titanate, which possess high mechanical and thermal strength, by B.M.Vul and I.M.Gel'dman (USSR) has played an important part in the study of ferroelectric materials and the extension of their fields of application. Barium titanate today is the most widely studied and most extensively used ferroelectric.

The use of ferroelectrics in digital computer technology is due to their ability of retaining residual polarization after removal of the external electric field.

Figure 142 shows the polarization of a crystal of barium titanate plotted against the external field. Let the residual polarization in the absence of a field be characterized by the point A of the hysteresis loop. If now an external field coinciding with the sense of the residual polarization is applied, the polarization of the crystal will increase along the line AB. If the field strength is then decreased, the polarization will decrease along the line BA. /25 For  $E = 0$ , the polarization of the crystal will be equal to the residual polarization.

Let us apply a field of opposite sense, gradually increasing in strength. The polarization will now decrease (segment AB). At a certain field strength  $E_c$ , the polarization will suddenly jump in the direction of the external field. The field strength  $E_c$  at which the sense of the polarization is reversed is called the coercive force. Further rise in field strength will lead to a smooth rise in polarization (segment DE). If now the field strength is decreased, the polarization will begin to decrease along the line EDG. At a field equal to zero, the state of the crystal will be characterized by the

point G of the hysteresis loop. The change in polarization when the field strength varies from  $+E_{max}$  to  $-E_{max}$  and from  $-E_{max}$  to  $+E_{max}$  constitutes the hysteresis loop.

Thus, in the absence of an external electric field, the state of the ferroelectric is characterized either by the point A or by the point G of the hysteresis loop. These states represent the binary digits 1 and 0.

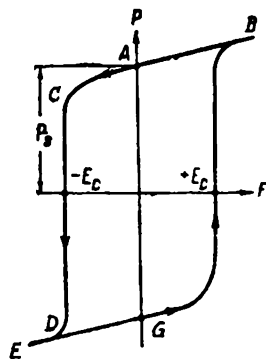


Fig.142 Polarization of a Barium Titanate Single Crystal as a Function of the Strength of the External Electric Field

Ferroelectrics find extensive use as bistable elements in various units of digital computers, especially in matrices of electrostatic memory units. They are beginning to be used as dielectric fillers for storage capacitors. Such capacitors with residual polarization also have two stable states of equilibrium, set by an external electric field. In a storage capacitor of small size, residual polarization and charges are stored for a long time without substantial changes even if the electrodes of the capacitor are short-circuited. This property is of great importance since it permits the use of simple switching systems and also makes regeneration unnecessary.

Figure 143 is the circuit of an elementary memory cell to store code 1 or 0, using the storage capacitor  $C_1$  with a ferroelectric filler. In writing the code 1, a voltage of one polarity is applied to the capacitor, whereas in writing the code 0, the voltage applied is of opposite polarity. Numbers are read by a voltage pulse of the same polarity used to record the code 1. The information is identified by the magnitude of the voltage from the capacitor  $C_2$  charged by the output currents. If a one was recorded, a large signal will 257 appear at the output, whereas if a zero was recorded, the output signal will be small.

Storage capacitors with a ferroelectric as dielectric can be used in the matrices of electrostatic memory units.

Matrix systems in which ferroelectric materials serve as the memory element can be manufactured by the use of printed circuits, which is a major ad-

vantage of such systems. Figure 144 shows part of a matrix-type memory unit consisting of a ferroelectric plate. On both sides of the plate, mutually perpendicular conducting strips x and y are applied by the printing method.

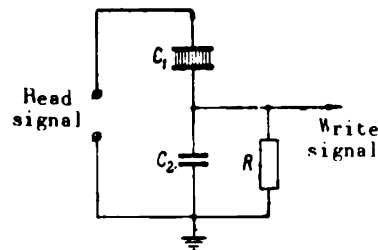


Fig.143 Circuit of Elementary Memory Cell Based on a Storage Capacitor with Ferroelectric Filler

The elementary memory cell is the region of the plate sandwiched between one of the x strips and one of the y strips. This constitutes a capacitor, with these

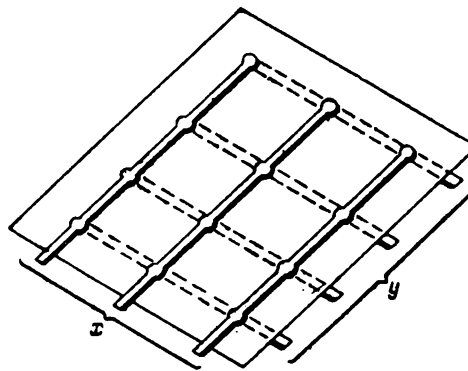


Fig.144 Matrix System Using a Ferroelectric Material as the Memory Element

areas of the x and y strips acting as plates and the ferroelectric acting as the dielectric filler. The operating principle is the same as that of a matrix of magnetic cores with a rectangular loop, except that here voltages are used instead of currents.

To record a binary digit in the elementary cell of a matrix, a voltage of one polarity is applied to one of the conducting x strips, and a potential of the same magnitude and opposite polarity to one of the y strips. The area of the ferroelectric between the two excited strips is polarized in the sense of the applied electric field, thus writing the binary digit to be stored. The other regions of the ferroelectric located along the excited x and y strips are subject to the action of an electric field with only half the voltage and, because of the rectangular polarization - field strength characteristic, remain

in practically the same state as before application of the field.

/258

In reading a one, the applied electric field is driven in a direction opposite to that of the field for writing a one. In reading a zero, the sense of the field is the same as in writing it. Consequently, a large signal will appear at the output of the circuit only when reading a one.

As an example, let us assume a ferroelectric matrix (a barium titanate plate) of an area of  $7.5 \text{ mm}^2$  and a thickness of  $0.125 - 0.25 \text{ mm}$ , having a capacity of 256 bits. Here, the plate has 16 electrodes each  $0.1 \text{ mm}$  wide, on one side of the surface, and 16 similar electrodes on the other side.

Ferroelectric memory units have the following advantages over memory units of the magnetic type:

Printed-circuit methods can be used in the manufacture of a matrix system.

The storage elements are very small.

Operation at low voltages is possible (below  $10 \text{ v}$ ), which is important in the design of computers using semiconductor diodes and triodes.

Work is now being done on the extensive introduction of ferroelectrics into digital computer technology. In particular a new ferroelectric - triglycine sulfate - has recently been developed in the United States. This compound is significantly superior to barium titanate, since hardly more than a tenth of the voltage is required to change its polarization, and the repolarization time is only  $1 - 2 \text{ } \mu\text{sec}$ .

## ARITHMETIC UNITS

Section 39. Arithmetic Units in General and their Principal Types

The solution of any mathematical problem on electronic digital computers reduces to a definite sequence of arithmetic and logic operations. The individual stages of the logic operations are very close to the corresponding stages of the performance of the arithmetic operations. This permits the performance of both arithmetic and logic operations by the same or similar elements, junctions, and building blocks, the ensemble of which constitutes the arithmetic unit of the computer. Thus, the arithmetic unit (AU) of a digital computer is an ensemble of elements, junctions, and modules designed to perform arithmetic and logic operations on numbers expressed in the selected system of notation and represented in the form of the appropriate codes.

An AU is based on building blocks and junctions designed to perform arithmetic operations: adders, multipliers, and dividers. An AU also includes registers, the logic elements AND, OR, NOT, OR-OR, and individual junctions to perform logic operations. According to the manner in which the elements and assemblies are used, arithmetic units are classified by structure as combination or universal (AU of combinatorial or universal, general-purpose type) and building block (AU of block type).

In a combination AU, most of the operations are performed by the same elements and assemblies which are not separated into individual building blocks. Such an AU is based on adders and registers interconnected in a definite way when performing the individual operations, by means of logic elements. The BESM-2 and "Ural-2", for instance, have arithmetic units of this kind. In block AU's, the principal arithmetic and logic operations are performed by /260 functional blocks that are practically independent of each other. Thus, the AU of the "Strela" computer has separate functional building blocks for adding and multiplication and logical building blocks for the logic operations.

The building blocks and assemblies of an AU that directly perform the arithmetic operations consist of various elements which customarily are classified into combinatorial or coincidence-type and accumulatory-type elements. The combinatorial elements are sometimes called code-positional, and the accumulatory, pulse-counting.

By type of elements, adders are divided into coincidence-type and accumulatory-type. Adders are also classified by method of input of the columns of the summands and the carry principle (the principle of arranging the transfer of carry ones).

Figure 145 indicates the classification of adders according to these



features.

Multipliers, like adders, can be either of the coincidence or accumulatory type. They are likewise classified by the method of input of the digits of the quotient (partial) products and the principle of adding the partial products. Figure 146 is a block diagram illustrating the classification of multipliers according to these features.

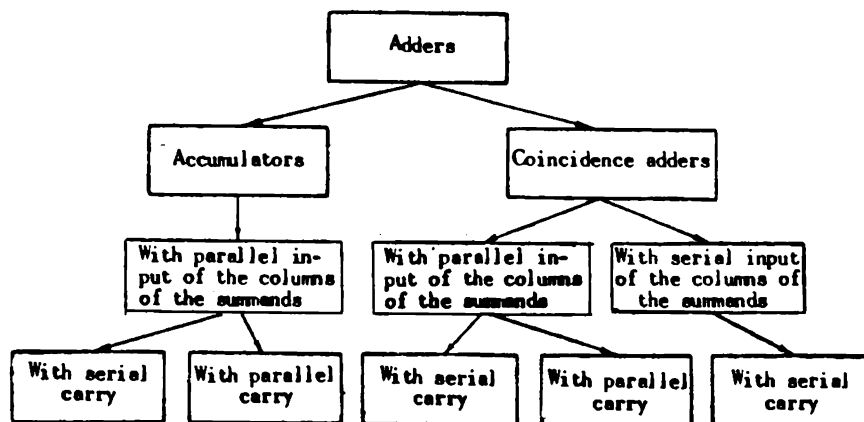


Fig.145 Classification of Adders

It must be noted that only multipliers of the coincidence type are used as individual building blocks in an AU. Multipliers of the accumulating type and similar dividers, as a rule, are formed in a combination AU by a definite pattern of switching the basic assemblies.

In an arithmetic unit, whatever its type, the main module or assembly is the adder, which may be constructed on one of these principles. The adder of an AU is composed of elementary one-column adders which come in innumeros 262 variants in their practical design.

One-column adders, like many-column adders, are divided into accumulating and coincidence-type. An accumulating one-column adder usually consists of a flip-flop with a counter input and a potential-pulse output. Coincidence bit adders are built of AND, OR, NOT, OR-OR logic elements, and two-input and three-input one-column adders are in existence.

#### Section 40. One-Column Adders

Numbers expressed in any system of notation are added column by column. For this reason, binary numbers can be added by the computer only if it has units that correctly add the digits of one column of the summands in any combination, taking account of the possible carry from the adjacent less significant column. One-column adders are such devices.

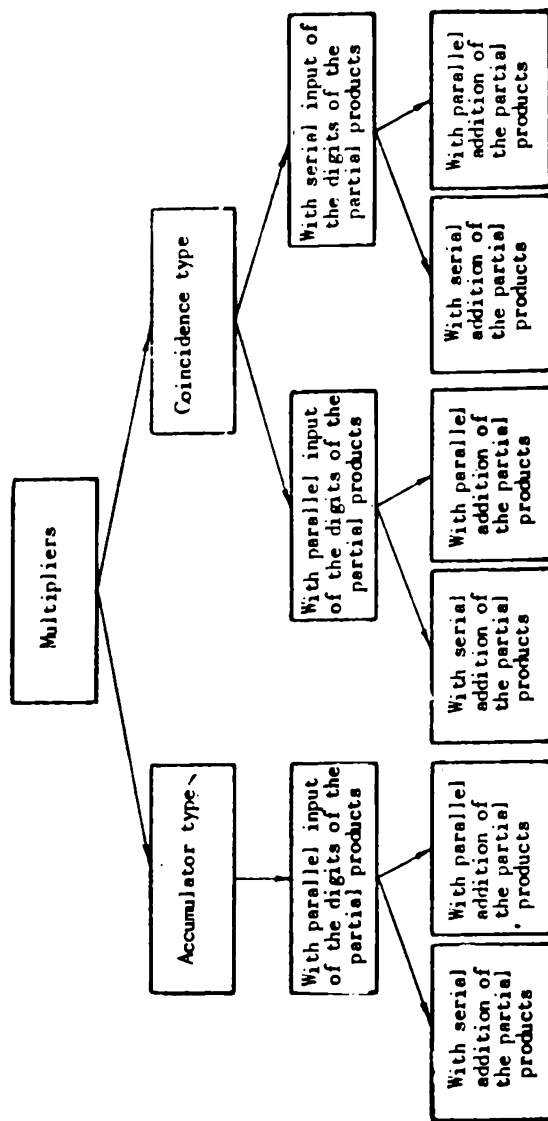


Fig.146 Classification of Multipliers

Two-input one-column adders. The two-input one-column adder (OA-2) is designed to add two one-digit binary numbers represented in the form of the corresponding pulses or potential levels.

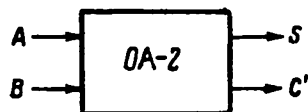


Fig.147 Block Diagram of the OA-2

The OA-2 has two inputs A and B and two outputs S and C' (Fig.147). When the digits to be added, represented in the form of pulses or potential levels, arrive at the inputs, the output S produces the digit of that particular column of the sum while the output C' furnishes the carry to the next higher column.

There can be four combinations of bits at the outputs of the OA-2. To these combinations must correspond definite combinations of digits at the outputs. Table 19 shows all these combinations.

TABLE 19  
LOGIC OPERATION OF THE OA-2

No.	Input A	Input B	Output of Sum S	Output of Carry C'
1	0	0	0	0
2	0	1	1	0
3	1	0	1	0
4	1	1	0	1

Writing the logic function for the output S according to the falseness /263 conditions, and for the output C' according to the truth conditions, we obtain the expressions

$$S = (A + B) (\bar{A} + \bar{B}),$$

$$S = (A + B) \bar{A}\bar{B} \text{ and } C' = AB.$$

These expressions are interpreted as follows:

A signal will appear at the output S if and only if there is a signal at one of the inputs (A + B) and no simultaneous signals at both inputs ( $\bar{A}\bar{B}$ ).

A signal will appear at the output  $C'$  if and only if there are simultaneous input signals at both inputs (AB).

Knowing the logical description of OA-2 operation, its circuit is readily laid out from elements performing the elementary logic operations. In fact, according to the above expressions, a one-column adder with two inputs can be designed from two logic AND circuits, one logic OR circuit, and one inverter (a NOT circuit).

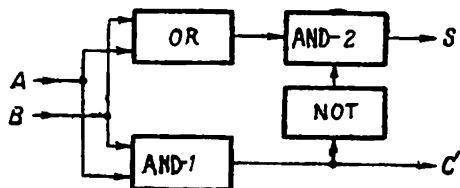


Fig.148 Function Circuit of the OA-2 (1<sup>st</sup> Version)

Figure 148 gives the function circuit of an OA-2 including an OR gate, two AND gates, and a NOT gate.

To check whether this adder operates correctly, let us feed its inputs with combinations of signals such that one will correspond to the presence of a signal and zero to its absence.

In the absence of signals at both inputs, there will be no changes in the state of the adder, and there will be no signals at either of its outputs (first row of the Table).

If a signal corresponding to the digit 1 is applied to the input A or to the input B (second and third rows of the Table), a signal will be formed at the output of the OR gate, and this signal then goes to one of the inputs of the AND-2 gate. In this case, there will be no signal at the output of the AND-1 gate, and the NOT gate will remain in its original state, in which it had opened the AND-2 gate. Thus, the signal from the output of the OR gate passes through the AND-2 gate to the output S. No signal will appear at the output  $C'$ , since there is no signal at the output of the AND-1 gate.

If signals are fed simultaneously to the inputs A and B (fourth row of the Table), then a carry signal will appear at the output of the AND-1 gate, and thus at the output  $C'$  of the adder. The same signal will compel the NOT gate to cut off the AND-2 gate, which then will not pass the signal formed at the output of the OR gate to the output S.

This circuit for a two-input one-column adder is not the only possible 264 one. The logic function of the output S is one of the modifications of the notation of the logic operation of equivalence negation, which is realized by an OR-OR gate. Thus by connecting an AND gate and an OR-OR gate in parallel to the inputs A and B, we obtain a second version of the function circuit of the

OA-2 (Fig.149). It is not difficult to prove that the operation of an adder built on this principle corresponds fully to the data of Table 19.

In practice, one-column adders with two inputs can be built from various components: electron tubes, ferrite cores, semiconductor diodes and triodes. Electron tubes as well as semiconductor diodes and triodes are usually employed to build an OA-2 according to the first logical version (see Fig.148).

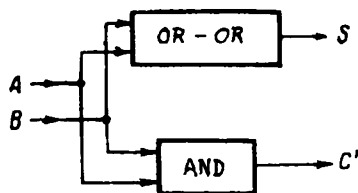


Fig.149 Function Circuit of the OA-2 (2<sup>nd</sup> Version)

To build an OA-2 according to the 2<sup>nd</sup> logical version (see Fig.149), it is more convenient to use ferrite-diode and ferrite-transistor cells, since the operation of equivalence negation can be more successfully realized in this case.

Vacuum-tube OA-2. It is difficult to build an OA-2 exclusively of electron tubes; therefore, semiconductor diodes are also introduced into this circuit. Figure 150 is a practical circuit of an OA-2 using electron tubes in combination with semiconductor diodes.

A high potential level at the adder inputs corresponds to code 0 and a low level to code 1. A low potential level at the outputs corresponds to code 0 while a high level corresponds to code 1. Thus, this OA-2 circuit is of the inverting type.

In the original state of the adder, when  $A = 0$  and  $B = 0$ , the vacuum tubes  $L_1$  and  $L_2$  are open, since the high potentials at the inputs compensate the action of the bias voltage  $-E_c$ . The tube  $L_3$  is cut off, since the bias voltage  $-E_c$  applied to its grid is not compensated at all. The diode  $D_1$  is cut off by the high plate potential of the tube  $L_3$ . Since the tubes  $L_1$  and  $L_2$  are open, current flows through the resistors  $R_1$ ,  $R_2$ , and  $R_5$  and the diodes  $D_4$  and  $D_5$ , and low potential levels are formed at the points M and N. Consequently, low potential levels appear also at the outputs S and C', i.e., the codes of zero are obtained there.

If the input A is fed the signal of code 1, i.e., a low potential, then the tube  $L_1$  is cut off while  $L_2$  remains open. The current will pass to the resistors  $R_2$ ,  $R_5$  and the diode  $D_5$ , i.e., a high voltage level will be formed at the point P and a low level at the point Q. The high potential of point P will be transmitted through the diode  $D_2$  and the resistor  $R_6$  to the output S, since the diode  $D_1$ , as before, is cut off by the high plate potential of the cutoff tube  $L_3$ . Thus, in this case, the signal of code 1 appears at the out-

put S and the signal of code 0 at the output C'.

/265

If a signal of code 1 is fed to the input B, then the adder will operate as in the case of application of the signal of code 1 to the input A, with the only difference that the high voltage level now appears at the point Q. This level will be transmitted to the output S through the diode  $D_3$  and resistor  $R_8$ .

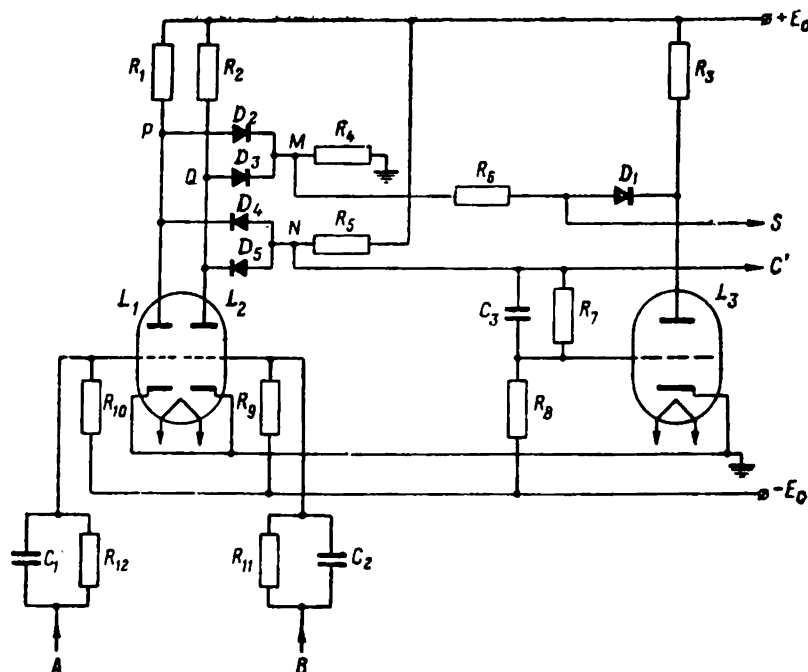


Fig.150 OA-2 with Electron Tubes and Semiconductor Diodes

When the signals of code 1 are applied at the same time to both inputs A and B, the tubes  $L_1$ ,  $L_2$  are cut off and a high potential level is created at the points M and N. The high potential of point N is transmitted to the output C' and the grid of tube  $L_3$ , which is opened. Since the tube  $L_3$  in this case is open, a low potential level will appear on its plate. The diode  $D_1$  is therefore opened and the high potential of the point M, passing through the tube  $L_3$  to the ground, is not transmitted to the output S. Thus the signal of code 1 appears at the carry output C', while the signal of code 0 appears at the sum output S.

In this adder, the diodes  $D_2$ ,  $D_3$  and  $D_4$ ,  $D_5$ , together with the tubes  $L_1$  and  $L_2$ , respectively, perform the functions of the logic elements OR and AND-1. The functions of the logic elements AND-2 and NOT are performed by the diode  $D_1$  and the tube  $L_3$ .

OA-2 with ferrite-transistor cells. The second logical version of the /266 OA-2 circuit uses one AND gate and one OR-OR gate, connected in parallel. Since the operation of inhibition used in the building of logic circuits with ferrite-

transistor cells can be realized in two ways, OA-2 circuits using these elements come in two versions.

In realizing the operation of inhibition by ferrite transistor cells with inhibit windings, the AND and OR-OR gates are constructed according to the following expressions:

$$S = (A + B) \overline{AB} = \overline{AB} + \overline{A}B;$$

$$C' = AB = A(\overline{\overline{AB}}).$$

Since the term  $\overline{AB}$  enters into the right-hand sides of both expressions, it follows that, in constructing an OA-2 by connecting AND and OR-OR gates in

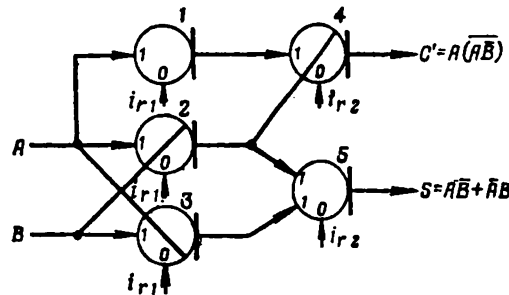


Fig.151 Structural Diagram of OA-2 with Ferrite-Transistor Cells Using Inhibit Windings

parallel, the ferrite-transistor cell is excluded in one of them, satisfying the relation  $\overline{AB}$ . On this account, the practical OA-2 gate does not contain six ferrite-transistor cells but only five, as shown in the structural diagram (Fig.151) and the wiring diagram (Fig.152).

The circuit shown in Fig.151 operates as follows: Initially, the ferrite cores of all cells are in the state 0. When the signal of code 1 arrives at the input A, it sets the ferrite cores of cells 1 and 2 into state 1. The state of cell 3 remains unchanged, however, since the input signal here arrives at its inhibit winding. The regular read pulse  $i_{r1}$  resets the ferrite cores to their original state, forming signals of the code 1 at the outputs of the corresponding cells.

The signal formed at the output of cell 2 sets the ferrite core of cell 5 into state 1. The ferrite core of cell 4 remains in state 0, since signals of code 1 from the outputs of cells 1 and 2 arrive simultaneously at its input winding and its inhibit winding. The next read pulse  $i_{r2}$  resets the ferrite core of cell 5 into state 0. A signal of code 1 is formed at the output S of the entire circuit. The signal of code 0 appears at the output C'.

If the signal of code 1 arrives at the input B, only the ferrite core of cell 3 is set into state 1 by it. The regular read pulse  $i_{r1}$  returns this /267

core to the original state, causing the appearance of the signal of code 1 at the output of cell 3 and setting the ferrite core of cell 5 into state 1. When the ferrite core of cell 5 is switched to state 0 by the next read pulse  $i_{r2}$ , a signal code 1 is formed at the output S. At the output C', in this case as well, there will be the signal of code 0.

When signals of code 1 are simultaneously fed to the inputs A and B, only the ferrite core of cell 1 is set into state 1, since the input signals compensate each other in cells 2 and 3. The next read pulse  $i_{r1}$  resets the ferrite core of cell 1 to state 0, giving the signal 1 at the output, and setting the ferrite core of cell 4 into state 1. The next read pulse  $i_{r2}$  resets the ferrite core of cell 4 to state 0. The signal code 1 is formed at the output C' but the signal of code 0 appears at the output S.

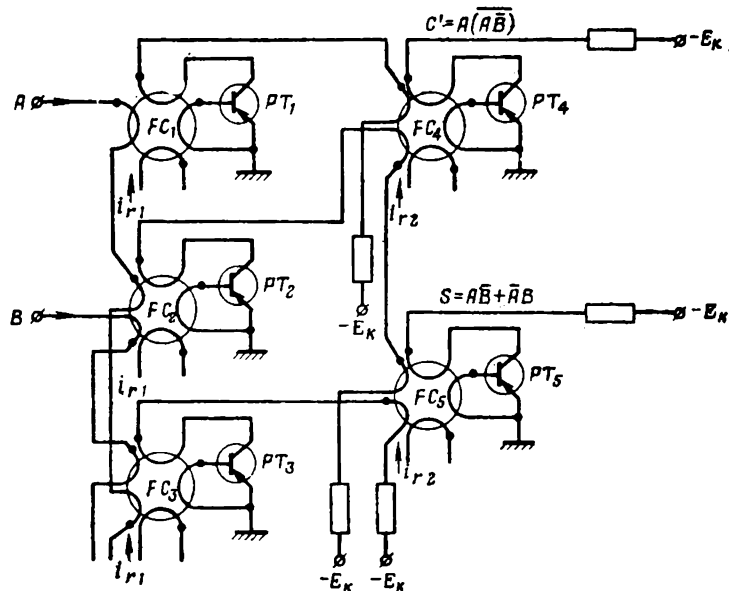


Fig.152 Wiring Diagram of the OA-2 Using Ferrite-Transistor Cells with Inhibit Windings

When the operation of inhibition is realized by the method of compensating the emf induced in the base winding, the two-input one-column adder can be constructed on one OR-OR gate. In fact, in this case the equivalent negation circuit realizes the function

$$S = (A + B) \overline{AB},$$

including the logical multiplication of the variables A and B, yielding a function of the output of the carry C'. Thus a two-input one-column adder, based on an OR-OR gate in which the operation of inhibition is realized by the method of compensating the emf of the base winding, is described by the following logical expressions:

/268



$$C' = AB;$$

$$S = (A + B) \overline{C'}.$$

Figure 153 is a wiring diagram of such an OA-2. It is obvious that the operation of this circuit is completely in accordance with the data of Table 19.

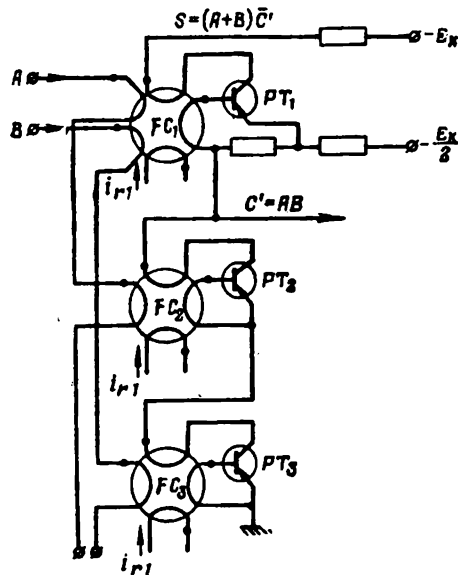


Fig.153 Wiring Diagram of the OA-2 with Ferrite-Transistor Cells, without Inhibit Windings

Three-input one-column adders. Such adders (OA-3) are designed to add three single-digit binary numbers (three bits) that simultaneously enter its

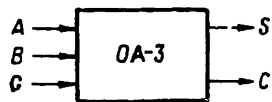


Fig.154 Block Diagram of the OA-3

inputs and that are represented in the form of the corresponding pulses or potential levels. This adder can perform all four elementary operations required for adding any corresponding columns of the summands.

The OA-3 has three inputs A, B, and C and two outputs S and C' (Fig.154). The digits of the given column arrive at the inputs A and B, and the carry from the adjacent less significant column at the input C. The outputs of the OA-3 are arranged like those of the OA-2; the digit of the given column of the sum

is produced at the output S, and the value of the carry to the next higher column at the output C'. As in the OA-2, some definite combination of digits at its outputs must correspond to every combination of bits at the adder inputs.

Table 20 gives all the combinations for the OA-3.

/269

TABLE 20  
LOGIC OPERATION OF THE OA-3

No.	A (1st Input)	B (2nd Input)	C (3rd Input)	S (Sum Output)	C' (Carry Output)
1	0	0	0	0	0
2	1	0	0	1	0
3	0	1	0	1	0
4	0	0	1	1	0
5	1	1	0	0	1
6	1	0	1	0	1
7	0	1	1	0	1
8	1	1	1	1	1

Writing the logic function for the outputs S and C' in accordance with the truth condition, we obtain the following expressions:

$$S = A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C + ABC;$$

$$C' = A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C + ABC.$$

To construct an OA-3 directly from the resultant expression requires a considerable number of AND, OR, and NOT gates. To reduce the pieces of hardware used in the adder, the logic relations realized by it are modified by identical transformations. In this case, the expression for C' can be substantially simplified, and the expression for S so transformed as to include all the expression for C'. The transformed expressions are written as follows:

$$S = ABC + (A + B + C)(\overline{AB + AC + BC});$$

$$C' = AB + AC + BC.$$

These logic expressions are interpreted as follows:

A signal will appear at the output S if and only if there are signals at all inputs (ABC) or at any one of them ( $A + B + C$ ) but no simultaneous signals at any two inputs ( $\overline{AB + AC + BC}$ ).

A signal will appear at the output C' if and only if there are simultaneous

1270

1270



1270

1270

1270



1270

1270

The one-column adder with three inputs can also be built up from OA-2 adders. To obtain such an adder, a second OA-2 must be connected in series with a two-input one-column adder, which we will denote as the "first OA-2" (see Fig.156). In addition, the OA-3 also contains an OR gate, which serves to /271 decouple the carry outputs of the two OA-2, to be combined into a single common carry output of the adder.

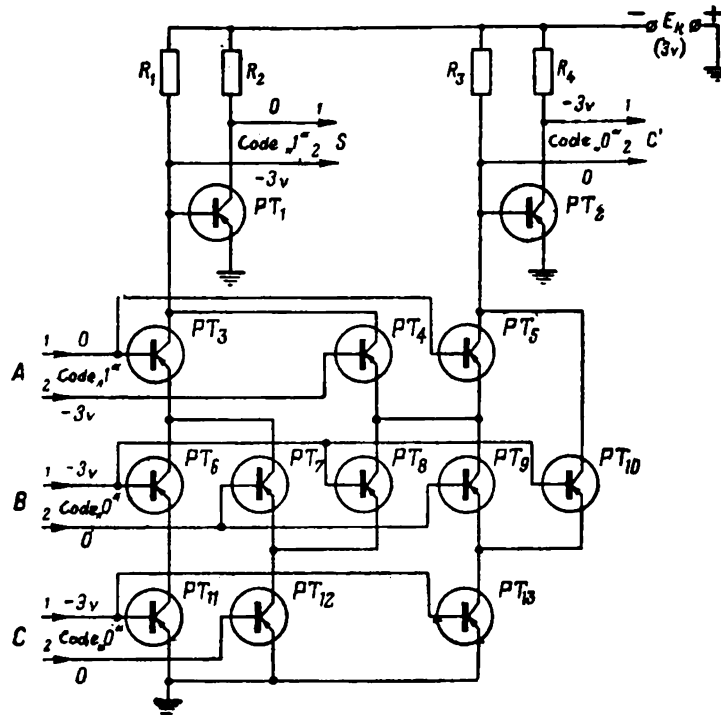


Fig.157 OA-3 with Semiconducting Triodes

It must be noted that the OA-2 forming a three-output one-column adder can be built in both the first and the second logical version.

In most practical cases, one-column adders with three inputs consist of two OA-2. Other OA-3 are also used, however, one of which is discussed below.

OA-3 with semiconducting triodes. A three-input one-column adder with semiconducting triodes has the simplest circuit in the case in which the triodes operate as control gates, i.e., are in the conducting or nonconducting state. Figure 157 is the circuit of such an OA-3. The adder consists of 13 junction transistors and four resistors and ensures complete correspondence of the input and output signal combinations. All the inputs and outputs of the adder have two buses each; a potential close to zero on the bus 1 at a potential of 3 v on the bus 2 corresponds to code 1, while a potential of -3 v on the bus 1 at a potential close to zero on the bus 2 corresponds to code 0.

The transistors  $PT_1$  and  $PT_2$  form the output. Depending on their state /272

(conducting or not), either code 1 or code 0 is formed at the outputs S and C'. The first buses of the outputs S and C' are coupled with the collectors of PT<sub>1</sub> and PT<sub>2</sub> respectively, so that a potential close to zero will form there when the transistors conduct current, i.e., when no current flows through the resistors R<sub>1</sub> and R<sub>3</sub> and when there is a potential of -3 v on the bases of these transistors. The second output buses are connected with the bases of PT<sub>1</sub> and PT<sub>2</sub> so that, when there is a potential close to zero on the first buses, a potential of -3 v is formed on the second buses.

Thus, when PT<sub>1</sub> or PT<sub>2</sub> conducts current, the potential on the bus of the outputs S or C' will correspond to code 1. When PT<sub>1</sub> or PT<sub>2</sub> does not conduct, the potential on the bus of the outputs S or C' corresponds to the code 0.

The triodes PT<sub>3</sub> - PT<sub>13</sub> are connected groupwise in series with the resistors R<sub>1</sub> and R<sub>3</sub>. The groups of transistors PT<sub>3</sub>, PT<sub>6</sub>, and PT<sub>11</sub>; PT<sub>3</sub>, PT<sub>7</sub>, and PT<sub>12</sub>; PT<sub>4</sub>, PT<sub>8</sub>, and PT<sub>12</sub>; PT<sub>4</sub>, PT<sub>9</sub>, and PT<sub>13</sub> are connected in series with the resistor R<sub>1</sub>; three groups of transistors PT<sub>5</sub>, PT<sub>8</sub>, and PT<sub>12</sub>; PT<sub>5</sub>, PT<sub>9</sub>, and PT<sub>13</sub>; PT<sub>10</sub> and PT<sub>13</sub> are connected in series with the resistor R<sub>3</sub>. Current flows through the resistor R<sub>1</sub> or R<sub>3</sub> only if all the transistors of one of the groups belonging to the given transistor are open. There is no current in these resistors when even a single triode in each group is cut off.

If the code 1 is applied to the input A of the adder, and the code 0 to the other inputs, then the potential on the buses of the inputs and outputs are distributed as shown in Fig.157. In this case, the potential on the buses of the output S corresponds to code 1, and on the buses of the output C' to code 0. This is due to the fact that some transistors (PT<sub>3</sub>, PT<sub>5</sub>, PT<sub>7</sub>, PT<sub>9</sub>, and PT<sub>12</sub>) of the adder are cut off while the others (PT<sub>4</sub>, PT<sub>6</sub>, PT<sub>8</sub>, PT<sub>10</sub>, PT<sub>11</sub>, and PT<sub>13</sub>) are open. One of each group of transistors belonging to the resistor R<sub>1</sub> is a cutoff triode. This means that no current flows through the resistor R<sub>1</sub>, that the triode PT<sub>1</sub> is open, and that the potential distribution on the buses of the output S corresponds to the code 1. At the same time, the transistors of one of the groups belonging to the resistor R<sub>3</sub>, namely, PT<sub>10</sub> and PT<sub>13</sub>, are open and current does flow through the resistor R<sub>3</sub>; the triode PT<sub>2</sub> is cut off, and the potential distribution on the buses of the output C' corresponds to code 0.

It is easy to prove that, in all other cases, the required correspondence of the input and output combinations of signals is obtained in this adder.

#### Section 41. Coincidence Adders

Coincidence adders are usually based on one-column adders with two or three inputs, corresponding in number to the method of input for the columns of the summands. For parallel input, the number of OA-2 and OA-3 is greatest, and for serial input it is smallest. In the former case, however, the speed /273 of the adding system is higher than in the latter.

Coincidence adders with parallel input of the columns of the summands and serial carry. Coincidence adders with parallel input of the columns of the

summands and serial carry can be designed with either OA-3 or OA-2. In these systems they are similar.

A three-input single-column adder is in most cases a complex combinatorial element. For this reason, an adder with parallel input of the columns of the summands using OA-3, is designed for the simultaneous input of both summands. The number of OA-3 in such an adder is determined by the maximum possible number of columns of the summands, and is equal to that number. Carry from the most significant  $n$ -th column is accomplished directly to the line of the  $(n + 1)$ -th column of the sum.

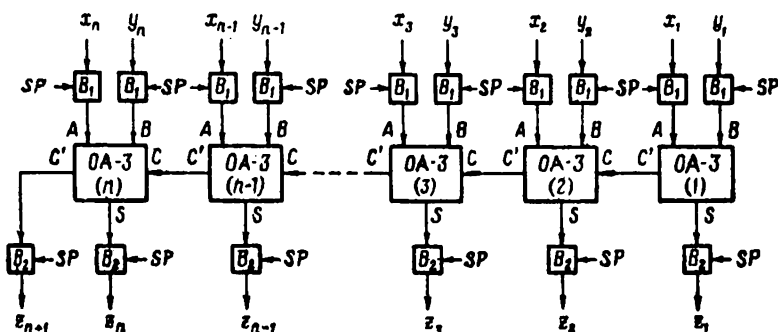


Fig.158 Coincidence Adder with Parallel Input of the Columns of the Summands and Serial Carry, Using OA-3

Figure 158 is a block diagram of such an adder, using OA-3 and designed to add two  $n$ -column binary numbers. According to this diagram, the adder has  $n$  (the number of columns of the summands) OA-3,  $n$  pairs of input gates  $B_1$ , and  $(n + 1)$  output gates  $B_2$ .

The input gates simultaneously deliver the code pulses of the summands to the inputs of the OA-3. The code pulses are synchronized by feeding sync pulses SP to the second inputs of the gates.

The output gates are used to synchronize the code pulses of the sum, which is also accomplished by feeding sync pulses SP to these gates.

The add elements in this system are the OA-3, whose inputs and outputs are distributed as follows: The code pulses of the given column of the summands are fed to the inputs A and B of each OA-3 from the outputs of the corresponding pair of gates  $B_1$ , and the carry pulses are driven to the input C. The output C' is used for the carry-out to the next higher column, and the output S <sup>[27]</sup> is the output at which the code pulse of the sum of the given column is formed.

If two  $n$ -digit binary numbers are being added in the adder

$$X = x_n x_{n-1} \dots x_{i+1} x_i x_{i-1} \dots x_3 x_2 x_1$$

and

$$Y = y_n y_{n-1} \dots y_{i+1} y_i y_{i-1} \dots y_3 y_2 y_1$$

where  $x_i$  and  $y_i$  are the digits of the  $i$ -th column of the summands, then the distribution of the code pulses corresponding to the digits of the summands at the inputs of the OA-3 will be as shown by Fig.158.

The input C of the OA-3 of the first column is not used, since no carry pulses pass into it. Thus, an OA-2 can be used in the first column of the adder instead of an OA-3.

In a coincidence adder using the circuits shown in Fig.158, all the pairs of digits of the same columns of the summands, represented in the form of code pulses, arrive simultaneously at the inputs of the corresponding OA-3 and are added simultaneously there. The resultant carries are transmitted to the OA-3 of the next higher columns, where they are taken into account in forming the final values of the digits of the sum. In the extreme case, a carry pulse appearing at the output  $C'$  of the OA-3 of the first column can be transmitted consecutively from one OA-3 to the next, including the OA-3 of the most significant  $n$ -th column.

The OA-3 as complex combinatorial elements are designed for the simultaneous arrival of all input pulses, including the carry pulse from the next lower place. The duration of the code pulses of the summands must therefore not be shorter than the maximum latency time of the carry pulse during its successive transfer from the OA-3 of the least significant column to the OA-3 of the most significant column.

In the two-input single-column adder (OA-2), two digits, represented by the corresponding code pulses, can be added. Since, in adding numbers, the possible carry from the next lower column must also be taken into account, an adder with parallel input of the columns of the summands needs two OA-2 for each column except the least significant one. Adders using OA-2 may have various systems, differing in the utilization of the inputs of the OA-2 composing the given column.

In one of these versions, the OA-2 that belong to a given column are assembled according to the conventional system, i.e., the inputs A and B of the first OA-2 are used to introduce the code pulses of the digits of the given column of the summands, and the carry pulse from the next lower column arrives at the input B of the second OA-2. Figure 159 gives a block diagram of an OA-2 adder using this version.

If two numbers  $X = x_n \dots x_3 x_2 x_1$  and  $Y = y_n \dots y_3 y_2 y_1$  are added and their 275 sum is  $Z = z_{n+1} z_n \dots z_3 z_2 z_1$ , then the distribution of the code pulses of the summands and the sum over the inputs and outputs of the OA-2 is as shown in Fig.159. Thus, the code pulses of the second column of the summands,  $x_2$  and  $y_2$ , are fed to the inputs  $A_1$  and  $B_1$  of the first OA-2 of the second column, marked OA-2<sub>1</sub> on the diagram. The output of the sum  $S_1$  of the OA-2<sub>1</sub> is connected to the input  $A_2$  of the second OA-2 marked OA-2<sub>2</sub> on the diagram. The carry pulse from the next lower column arrives at the input  $B_2$  of the OA-2<sub>2</sub>. The carry outputs  $C'_1$  and  $C'_2$  of both OA-2 are combined by means of an OR gate into a common carry output  $C'$  of the given column. The code pulse of the second column of the sum  $z_2$  is formed at the output of the sum  $S_2$  of the OA-2<sub>2</sub>.

A parallel adder based on OA-2 and designed according to this version of the system, operates in the following sequence: First the digits of the summands are added column by column and then the carries from the next lower columns are added to the digits of the resultant sum. This sequence gives maximum speed in serial carries.

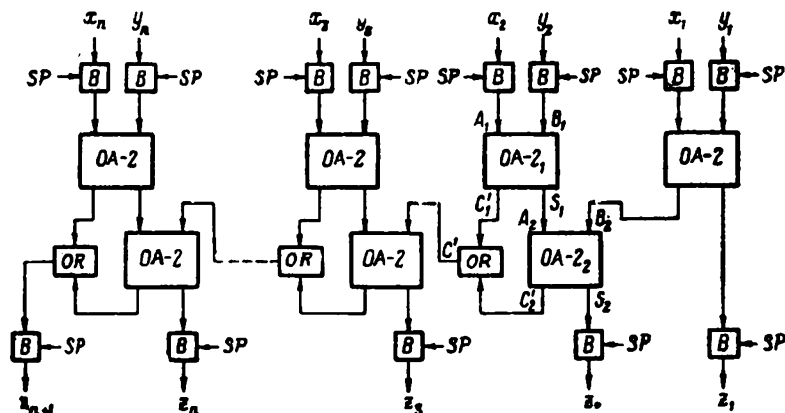


Fig.159 Coincidence Adder with Parallel Input of Columns of the Summands and Serial Carry, Using OA-2

Coincidence adder with parallel input of the columns of the summands and parallel carry. Parallel carry can be accomplished in a coincidence adder designed on the basis of OA-2. For this purpose two OA-2 are connected in each column of the adder except the least significant: One of these OA-2 adds the digits of the given column while the other forms the actual value of the same column of the sum. The adder also includes gates forming the parallel carry.

Figure 160 shows the circuit of a coincidence adder with parallel input of the columns of the summands and parallel carry, designed to add two four-digit binary numbers. In adding two numbers  $X = x_4x_3x_2x_1$  and  $Y = y_4y_3y_2y_1$ , whose sum is  $Z = z_5z_4z_3z_2z_1$ , the distribution of code pulses of the summands and the sum over the inputs and outputs of the adder is shown in Fig.160. /276

The code pulses of the summands are fed to the inputs of the first stage of the one-column adders, which are the adders OA-2<sub>1</sub>, OA-2<sub>2</sub>, OA-2<sub>3</sub>, and OA-2<sub>4</sub>. The carry outputs of these OA-2 are connected by means of the gates B<sub>1</sub>, B<sub>2</sub>, and B<sub>3</sub> with a parallel carry loop to the individual points to which the inputs of the OA-2 of the second stage are connected. The code pulses formed at the sum outputs of the OA-2 of the first stage drive the gates of the parallel carry line and are also fed to the inputs of the OA-2 of the second stage.

The second stage of the one-column adders is formed by the adders OA-2<sub>5</sub>, OA-2<sub>6</sub>, and OA-2<sub>7</sub>. These OA-2 are designed to form the actual values of the columns of the sum without taking account of the secondary carry pulses, which is what is required in adders with parallel carry. The secondary carry pulses



are not taken into account because the carry outputs of the second stage of the OA-2 are not used.

Consider the operation of the adder elements in adding the binary numbers

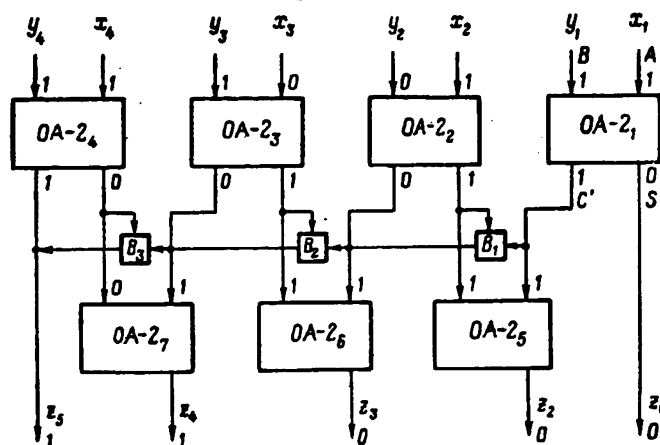


Fig.160 Coincidence Adder with Parallel Input of the Columns of the Summands and Parallel Carry, Using OA-2

$X = 1011$  and  $Y = 1101$ . When the code pulses of these numbers are applied to the inputs of the OA-2 of the first stage, the carry pulses are formed at the outputs of OA-2<sub>1</sub> and OA-2<sub>4</sub>. The carry pulse from the output of the OA-2<sub>4</sub> goes directly to the loop of the fifth column of the sum, and the carry pulse from the output of the OA-2<sub>1</sub> to the parallel carry. Since the pulses of code 1 /277 were formed at the sum outputs of the OA-2<sub>2</sub> and OA-2<sub>3</sub>, the carry pulse passes from the output of the OA-2 through the gates B<sub>1</sub> and B<sub>2</sub> and arrives at the inputs of all the OA-2 of the second stage. Pulses of the code 1 also arrive at the second inputs of the OA-2<sub>5</sub> and OA-2<sub>6</sub> from the sum outputs of the OA-2<sub>2</sub> and OA-2<sub>3</sub>. For this reason, the code signals 0 appear in the output loops of the second and third columns of the sum, as in the first-column loop. The code signal 0 arrives at the second input of the OA-2<sub>7</sub>, and for this reason the code signal 1 is obtained at its output, i.e., in the loop of the fourth column of the sum.

Thus, as a result of the addition, the number  $Z = 11\ 000$  which equals the sum of  $X$  and  $Y$  is formed.

The carry pulse that arises during addition in any column of a coincidence adder using this particular circuit, as it advances toward the more significant columns, is delayed only in the gates. Since this delay is usually substantially less than the delays in the OA-2 or OA-3, coincidence adders with parallel carry are faster than coincidence adders with serial carry.

Coincidence adders with serial input of the columns of the summands.

These adders may be built either of OA-3, directly set up from the elementary logic circuits, or of OA-2 and additional delay lines.

The circuit of a coincidence adder using serial input of the columns of the summands, based on an OA-3 directly composed of elementary logic circuits, is shown in Fig.161. The inputs A and B of the OA-3 are used for serial input

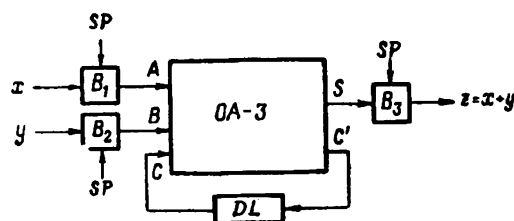


Fig.161 Coincidence Adder with Serial Input of the Places of the Summands Based on OA-3

of the columns of the summands. The input C of the OA-3 is connected through a delay line with its carry output C', and is used to transfer the carry codes from the lower columns to the next higher columns.

The delay line DL in the carry line is so designed that the code carry pulse, formed when the digits of the m-th column of the summands are added, will arrive at the input C simultaneously with the arrival at the inputs A and B of the code pulses of the digits of the (m + 1)-th column. If  $t_c$  is the pulse delay time in the OA-3, then the pulse delay time  $t_d$  in the delay line (DL) will be

$$t_d = \frac{1}{f} - t_c,$$

where  $f$  is the repetition rate of the code pulses of the summands.

Here,  $\frac{1}{f}$  is the time interval between two successive code pulses of /278

the numbers; it is usually called the clock time and is denoted by the letter T.

The arrival of the code pulses of the summands at the inputs A and B is synchronized by the gates  $B_1$  and  $B_2$ , which control the sync pulses CP. The repetition rate of the sync pulses equals the major cycle frequency of the computer. The output gate  $B_3$  is used to synchronize the code pulses of the sum at the output S of the OA-3. The sum, like the summands, is represented in the form of a serial pulse code. The codes of the sum and summands are transmitted with the least significant columns first.

The adding time for two numbers on a coincidence adder designed on the basis of the above circuits may be calculated as follows: The formation of the sum is considered completed at the instant the last code pulse of the sum appears at the output of the adder. If the first code pulses of the summands

were fed to the adder inputs at the time  $t_0$ , then the first code pulse of the sum will appear at the output at the time  $t_0 + t_c$ . With  $n$ -column summands, the sum can be an  $(n + 1)$ -column number. Consequently, the last code pulse of the sum appears at the adder output at the time  $t_0 + t_c + nT$ .

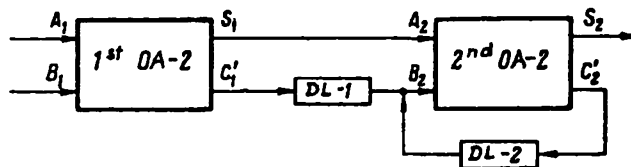


Fig.162 Coincidence Adder with Serial Input of the Columns of these Summands Using Two OA-2 (1<sup>st</sup> Version)

It follows from the above reasoning that the adding time of two  $n$ -column binary numbers in an adder using the circuit of Fig.161 is

$$T_z = nT + t_c$$

The delay time of the pulse  $t_c$  in a vacuum-tube adder is often far less than the time of a single clock  $T$ , permitting us to consider that  $T_z = nT$ . If the operating frequency of such an adder is 200 kc, then the time it will require to add two 18-column numbers will be

$$T_z = \frac{1}{200000} \cdot 18 \text{ sec} = 90 \text{ } \mu\text{sec.}$$

Many practical coincidence adders with serial input of the digits to be added use two OA-2 in whose carry loops delay lines are connected. Figure 162 shows a general circuit for such an adder. The summands, represented in the 279 form of serial code pulses, are fed, the least significant columns first, to the inputs  $A_1$  and  $B_1$  of the first OA-2. The serial pulse code of the sum is formed at the output  $S_2$  of the second OA-2.

The carry to the next higher column may occur either during addition of the digits of a given column or on addition of three digits of the given column of one of the summands with a carry 1 from the next lower column. In either case, the arrival of the carry pulse at the input  $B_2$  of the second OA-2 occurs simultaneously with the arrival of the code pulse from the output  $S_1$ , at the input  $A_2$  of the same OA-2. The necessity of synchronizing the arrival of the pulses at the inputs  $A_2$  and  $B_2$  is the starting point for calculating the pulse delay time on the delay lines DL-1 and DL-2.

Owing to the connection of an OA-2 in such a coincidence adder, the output pulses arise a certain time after the input pulses, and these output pulses are nonsimultaneous at the different outputs. The length of the pulse delay time in the OA-2 depends primarily on the characteristics of its elements and may vary from a value of close to zero to  $2T$  ( $T$  is the period of the code

pulses).

The dependence of the delay line characteristics on the characteristics of the OA-2 in the adder circuit is established under the supposition that the two OA-2 are the same, as is usually the case in practice. The OA-2 can be represented as a device with two channels, the input-output channel of the sum  $S$  and the input-output channel of the carry  $C'$ . Let us denote these channels by AS and BC', respectively. Let us also denote the pulse latency time in the channel AS by  $t_1$ , in the channel BC' by  $t_2$ , in DL-1 by  $t_3$ , and in DL-2 by  $t_4$ .

The first code pulse of the sum in this coincidence adder is delayed in the channels  $A_1S_1$  and  $A_2S_2$ . The pulse will appear at the output  $S_2$  lagging by the time  $2t_1$  with respect to the arrival of the first code pulses of the summands at the inputs  $A_1$  and  $B_1$ .

If a carry pulse is formed at the output  $C'$ , then it must arrive at the input  $B_2$  of the second OA-2 simultaneously with the arrival of the regular pulse from the output  $S_1$  of the first OA-2 at the input  $A_2$ . This condition can be satisfied only if the equality

$$t_1 + T = t_2 + t_3,$$

is valid, whence it follows that the pulse delay time in the DL-1 will be

$$t_3 = T + (t_1 - t_2).$$

Bearing in mind that the pulses arriving at the inputs  $A_2$  and  $B_2$  on formation of a carry pulse at the output  $C'_2$  must be coordinated in time, it is easy to obtain the relation

$$T = t_2 + t_4$$

or

$$t_4 = T - t_2$$

/280

It is clear from a comparison of the expressions for the pulse delay time on DL-1 and DL-2 that, in all cases,

$$t_3 > t_4,$$

and that their difference is constant and always equal to  $t_1$ , i.e.,

$$t_3 - t_4 = t_1.$$

Since always  $t_3 > t_4$ , the delay lines DL-1 and DL-2 can be combined into a single common delay line consisting of two segments connected in series. Its second segment will be none other than the DL-2, while its first segment is a delay line with the latency time  $t_1$ . Figure 163 gives a general wiring diagram of a coincidence adder with combined delay line.

The adding time  $T_{\Sigma}$  of two  $n$ -digit binary numbers in such a coincidence adder composed of two OA-2 is calculated by the formula

$$T_{\Sigma} = nT + 2t_1.$$

Here,  $T_{\Sigma}$  denotes the time between the arrival of the code pulses of the lowest columns of the summands at the adder inputs and the time of formation of

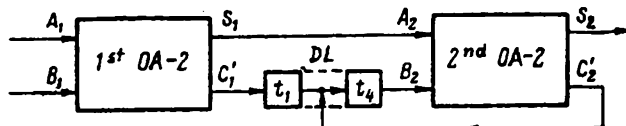


Fig.163 Coincidence Adder with Serial Input of the Columns of the Summands, Using Two OA-2 (2nd Version)

the code pulse of the highest column of the sum at the adder outputs.

The relation between the pulse delay time in various channels of the OA-2 and on the delay line is taken into account in building specific adder circuits. The use of these relations permits the rapid and accurate build-up of a coin-

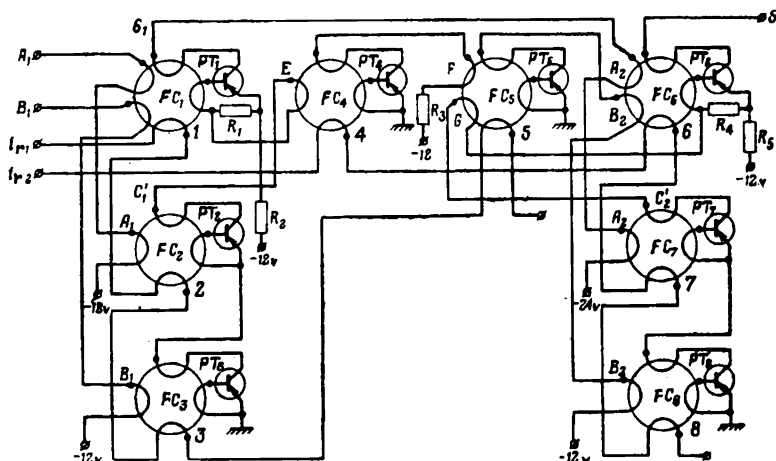


Fig.164 Schematic Diagram of Ferrite-Transistor-Cell Coincidence Adder

cidence adder with serial input, based on OA-2 of the given specific type.

Consider as an example, an adder with serial input of the columns of the summands, using OA-2 and ferrite-transistor cells without inhibit windings. In such OA-2 (cf. Fig.163) the channels AS and BC' are characterized by the fact that each of them is composed of a single ferrite-transistor cell. For this

reason, they have the following time characteristics:

/281

$$t_1 = \frac{T}{2} \text{ and } t_2 = \frac{T}{2}.$$

Substituting these values of  $t_1$  and  $t_2$  in the formula for the time characteristics of the delay lines used in an adder, we get

and

$$t_3 = T + \left( \frac{T}{2} - \frac{T}{2} \right) = T$$

$$t_4 = T - \frac{T}{2} = \frac{T}{2}.$$

In this case, it is advisable to use a common delay line, i.e., to build the serial adder according to Fig.163. In fact, here

$$t_1 = t_3 - t_4 = \frac{T}{2}$$

and each of the two parts of the common delay line can be a single ferrite-transistor cell.

The adding time for two n-digit binary numbers on the adder under discussion is

$$T_z = nT + 2\frac{T}{2} = (n+1)T$$

Figure 164 is a schematic diagram of this serial coincidence adder, using ferrite-transistor cells without inhibit windings. It consists of eight ferrite-transistor cells: the cells 1, 2, and 3 form the first OA-2; the cells 6, 7, and 8 the second OA-2; the cells 4 and 5 the common delay line. The input and output code pulses are synchronized by the read pulses  $i_{r2}$ . /282

The operation of this serial coincidence adder will now be considered from the example of addition of the two binary numbers 111111 and 110101. The addition of these numbers, represented by serial pulse codes and fed with the least significant places first, corresponds to the timing chart of the adder operation shown in Fig.165.

When a pulse of the series  $i_{r2}$  passes through the circuits of the read windings, the code pulses of the lowest columns of the summands are fed to the inputs  $A_1$  and  $B_1$  and set the ferrite cores  $FC_1$ ,  $FC_2$  and  $FC_3$  to the state 1. The next pulse of the series  $i_{r1}$  resets them to the state 0, causing the production, at the output  $C_1$  and thus also on the input E of the common delay line, of a pulse that sets the ferrite core  $FC_4$  to the state 1. When the next pulse of the series  $i_{r2}$  passes, the code pulse 1 is fed only to the input  $A_1$ , thus setting the ferrite cores  $FC_1$  and  $FC_2$  to the state 1. At this same time, the ferrite core  $FC_4$  is reset by the read pulse to the state 0 and a pulse enters the input F of the second cell of the common delay line, setting the core  $FC_5$  /283 to the state 1.

The next regular read pulse of the series  $i_{r,1}$  resets  $FC_1$ ,  $FC_2$ , and  $FC_5$  to the state 0, thus causing the appearance, at the inputs  $A_2$  and  $B_2$  of the second OA-2, of pulses setting  $FC_6$ ,  $FC_7$  and  $FC_8$  to the state 1. When the next read pulse of the series  $i_{r,2}$  passes, the codes 1 are recorded in the cells 1, 2, and 3; at the same time, the codes of 1 are read out of cells 6, 7, and 8, meaning that  $FC_6$ ,  $FC_7$ , and  $FC_8$  are reset to the state 0. A pulse is formed at the output  $C'_2$  and enters the input G of the second cell of the delay line,

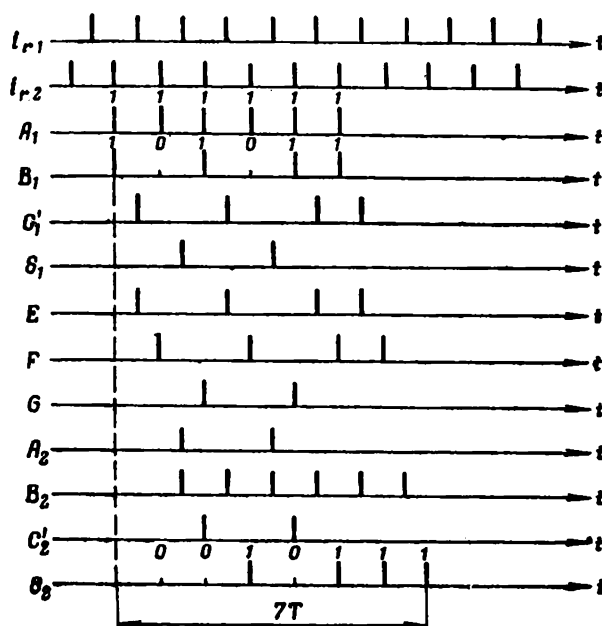


Fig.165 Timing Chart of Adder Operation

setting  $FC_5$  to the state 1. No pulse of code 1 is formed at the output  $S_2$ , i.e., on the common output of the sum of the whole adder; this corresponds to the digit 0 in the least significant column of the sum.

The further column-by-column addition of the numbers 111111 and 110101 proceeds similarly, completely in accordance with the timing chart in Fig.165. According to this chart, the adding time in this case is  $7T$ , which is in complete agreement with the data obtained by calculation based on the above formula.

Coincidence adders with serial inputs of the columns of the summands operate directly with the same number codes that arrive at their inputs. Such adders contain no special devices for taking care of the signs of the summands, and therefore the arithmetic sum is formed at their output. To obtain the algebraic sum, the complement code of the negative number must first be formed and the summands must be fed to the adder in such a manner that the code pulses of the signs directly follow the code pulses of the digits of the most significant columns of the summands; when this is done, the carry from the sign column must

be neglected.

## Section 42. Decimal Adders

Decimal adders are used in digital computers that operate with numbers represented in the decimal system. The coding of decimal digits may vary widely. Since bipositional, binary elements are most often used in practice, the digits of decimal adders are in most cases coded by various combinations of binary digits.

The simplest method of coding decimal digits by binary digits is the use of a binary-decimal code or binary-coded decimal system of notation. Each decimal digit is coded by the binary tetrad numerically equal to it, i.e., by the corresponding four-digit binary number. The binary-decimal code is sometimes called the 8421 code, from the values of one in the four places of a binary tetrad.

A decimal adder with a full number of digits, like a binary adder, can /284 be constructed with devices adding a single column, i.e., adding two decimal

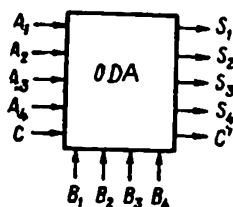


Fig.166 Block Diagram of ODA

digits (two one-digit decimal numbers). Such devices, which are called one-column decimal adders, consist of one-column binary adders OA-2 and OA-3, and also of AND, OR, and NOT logic gates.

When the binary-decimal code is used, a one-column decimal adder (ODA) is a piece of hardware with nine inputs and five outputs (Fig.166). The binary code of one of the decimal digits being added is fed to the inputs  $A_1 - A_4$ , the binary code of the second decimal digit to the inputs  $B_1 - B_4$ , and the carry from the next lower decimal column to the input  $C$ . The binary code of the decimal digit of the given column of the sum is formed at the outputs  $S_1 - S_4$ , and the carry to the next higher decimal column at the output  $C'$ .

The function circuit of the ODA can be constructed from the logical expressions obtained from the operation table, containing all possible combinations of input signals and the corresponding combinations of output signals. Such a design of ODA circuits, however, is very complicated since at least 200 possible combinations of input signals must be taken into account. The function circuit of the ODA is therefore usually directly built up, taking account of all peculiar features of the addition of decimal digits represented by binary



tetrads in the 8421 code.

One of the special features of addition in the 8421 code is that the formation of a carry in the next higher column depends on the value of the sum of the original decimal digits. The following three cases are possible.

First case: The sum of the decimal digits does not exceed eleven.

In this case, the value of the carry to the next higher column is determined from the values of the second and fourth (counting from right to left) columns of the binary tetrad representing the sum of the original digits. In fact, the carry to the next higher column must proceed at values ten and eleven for the sum. Here the second and fourth columns of the binary tetrad of the sum necessarily contain ones.

Examples:

1) 0100 - four	2) 0101 - five
+ 0110 - six	+ 0110 - six
<hr/> 1010 - ten	<hr/> 1011 - eleven

Second case: The sum of the decimal digits is from twelve to fifteen.

In this case, the value of the carry to the next higher column is determined from the values of the third and fourth columns (counting from right to left) of the binary tetrad representing the sum, since the presence of ones in these columns indicates that the sum is greater than ten. /285

Examples:

1) 0110 - six	2) 0110 - six
+ 0110 - six	+ 1001 - nine
<hr/> 1100 - twelve	<hr/> 1111 - fifteen

Third case: The sum of the decimal digits is greater than fifteen.

In this case, the carry from the highest column of the binary tetrads occurs on their addition since the decimal numbers sixteen, seventeen, eighteen are represented by five-digit binary numbers. The presence of this carry will be the indication that there is a carry to the next higher decimal column.

Thus, when adding in the 8421 code, the carry ones must be formed in the next higher decimal column when the sums in the second and fourth or the third and fourth columns of the binary tetrad contain ones, or when there is a carry from the most significant column of the binary tetrads on their addition.

The second feature of addition in the 8421 code is that, on a carry to the next higher decimal column, the value of the sum must be corrected.

When the sum of the decimal digits is greater than fifteen, i.e., in the case of a carry from the most significant column of the binary tetrads being



One variant of the function circuit of this adder is given in Fig.167. Here, the ODA contains five OA-3, two OA-2, two logic AND circuits and one OR circuit.

The one-column binary adders OA-3<sub>1</sub> - OA-3<sub>4</sub> serve to add the original binary tetrads, taking account of the carry from the next lower decimal column. By means of the one-column binary adders OA-3<sub>5</sub>, OA-2<sub>1</sub>, and OA-2<sub>2</sub>, the original value of the sum obtained at the outputs of the adders OA-3<sub>1</sub> - OA-3<sub>4</sub> is corrected by adding the number six to it. In this case, the binary code of the number six is formed by feeding the code signal 1 to the inputs OA-2<sub>1</sub> and OA-3<sub>5</sub> from the carry output C'.

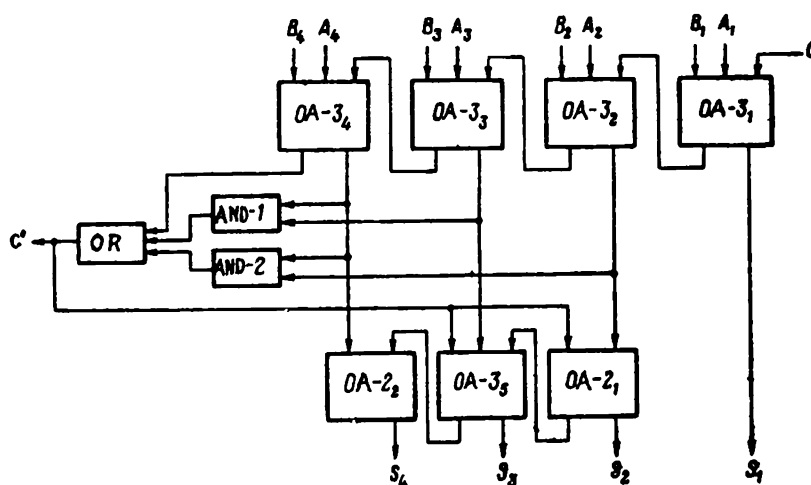


Fig.167 Function Circuit of the ODA for 8421 Code

The AND-1, AND-2 and OR logic circuits are designed to form the carry to the next higher decimal column. The AND-1 and AND-2 gates are connected to the outputs of the OA-3<sub>4</sub>, OA-3<sub>3</sub> and OA-3<sub>4</sub>, OA-3<sub>2</sub> respectively. For this reason, when the code signals 1 appear at the outputs of these pairs of OA-3, there is a carry signal at the output of AND-1 or AND-2. The OR circuit combines the circuits of carry formation into the single carry output C' of the adder.

In this variant of the ODA, the binary tetrads representing one-digit 287 decimal numbers are always correctly added. This is explained by the fact that all the peculiarities of addition in the 8421 code are taken care of in this model. The carry from the most significant column of the binary tetrads is neglected in correcting the value of the original sum, since the carry output of the OA-2<sub>2</sub> is not used.

Full decimal adders may have either parallel or serial input of the binary tetrads representing the decimal digits of the summands. In both cases, the circuits of the decimal adders are similar to the corresponding circuits of the binary adders based on the OA-2 and OA-3.

### Section 43. Coincidence-Type Multiplication Units

Coincidence-type multipliers can be designed for either serial or parallel input of the columns of partial products. In both cases, the partial products can be added either serially or parallel.

The most frequently used coincidence multipliers are of three types: with serial input of the columns of the partial products and serial addition of the partial products; with serial input of the columns of the partial products and parallel addition of the partial products; with parallel input of the columns of the partial products and parallel addition of the partial products.

Coincidence-type multiplication units with serial input of the columns of the partial products and serial addition of the partial products. These multipliers are used where the binary multiplicands can be represented only by 288 serial pulse codes. Shift registers or dynamic registers are used to store and shift the numbers in such multipliers. The partial products are formed by means of AND gates and the addition by means of coincidence adders with serial

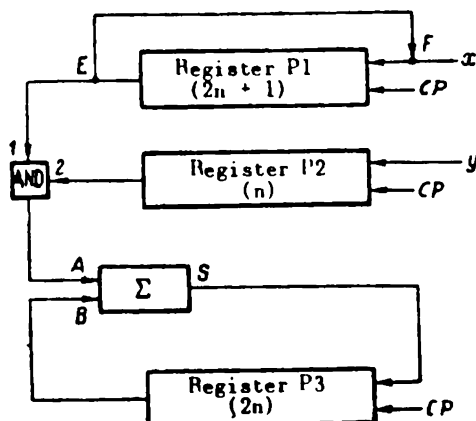


Fig.168 Coincidence-Type Multiplier with Serial Addition of Partial Products

input of the columns of the summands.

Figure 168 is a block diagram of one version of this multiplier. Here, the multiplier consists of a multiplicand register P1, a multiplier register P2, a product register P3, a coincidence adder with serial input of the summand columns  $\Sigma$  and an AND gate. The adder  $\Sigma$  and the register P3 can be regarded as an accumulative adding circuit of sequential action.

The registers have various numbers of columns, permitting automatic shifting of the partial products by one column for adding. If the multiplier is designed to take  $n$ -digit numbers, then the register P3 as the product register has  $2n$  columns. The number of columns in the register P1 depends on the number

of columns in the register P3 and on the pulse delay time in the adder and the AND gate. To simplify our further discussion of the operation of the multiplier, we shall consider that the pulses pass through the AND gate and the adder without delay. In this case, the register P1 has  $2n + 1$  columns, as shown in Fig.168. The register P2 does not directly participate in the mutual shifting of the partial products, for which reason the number of columns in it is  $n$  (the number required to take the digits of the multiplier).

Before multiplication, the respective values of the multiplicand and multiplier must be introduced into the registers P1 and P2. They are applied in the form of serial pulse codes, the less significant columns first. The more significant columns of the numbers are recorded in the rightmost columns of the registers.

The initial stage of multiplication is the formation of the first partial product. Like all the subsequent partial products, it is formed by the aid of an AND gate, for which purpose all the digits of the multiplicand, commencing with the least significant, are fed consecutively to its input 1, and the first column of the multiplier is fed  $2n + 1$  times to the input 2. /289

The principle of feeding the code pulses of the columns of the multiplicand and multiplier to the inputs of the AND gate depends on the type of elements used in the multiplier. If ferrite-transistor cells are used, then the code pulses of the multiplicand are fed to the AND gate by shifting the multiplicand in the P1 register during continuous feeding of the clock pulses. The multiple input of the code pulses of the first column of the multiplier to the input 2 of the AND gate is accomplished by having this pulse excite the controlled generating cell whose output is connected to the input 2 of the AND gate.

The first partial product from the output of the AND gate goes to the input A of the adder and then to the input of the register P3, where it is successively shifted from right to left. During the same time, the multiplicand is rewritten by the feedback circuit EF into the register P1. By the end of the first stage of operation of the multiplication unit, the multiplicand is again recorded in the register P1, while the first partial product is recorded in the right half of the register P3. Clock or shift pulses are fed continuously to record and shift the numbers in the registers P1 and P3 (these pulses in Fig.168 are the control pulses CP).

In the second stage of operation of the unit, the second partial product is formed and added to the first. Since the register P3 has one column less than the register P1, the first partial product arrives one beat before the second at the adder input. The sum of the first two partial products goes from the adder output to the register P3.

During the third stage of operation of the unit, the third partial product is formed and added to the sum of the first two partial products, which was obtained in the second stage, and so on. By the end of the last or  $n$ -th stage of operation of the unit, the sum of all the partial products is set up in the register, i.e., the value of the product of the numbers to be multiplied.

Each operating stage of the unit requires  $2n + 1$  clocks; the total number of stages is  $n$ . One more clock is necessary in addition to record, in the register P3, the code pulse of the most significant column of the final product, which is formed as a carry pulse in the adder system; the product now completely fills the register P3. The time required to multiply two  $n$ -digit binary numbers by means of a coincidence-type multiplier with serial input of the columns of the partial products and serial addition of the partial products is, therefore,

$$T_y = [(2n + 1)n + 1] T,$$

where  $T$  is a single clock time.

This time can be shortened by modifying the circuit of the unit. The number of columns of the multiplicand register is decreased by dividing the product register into two subregisters and introducing two additional logic /290

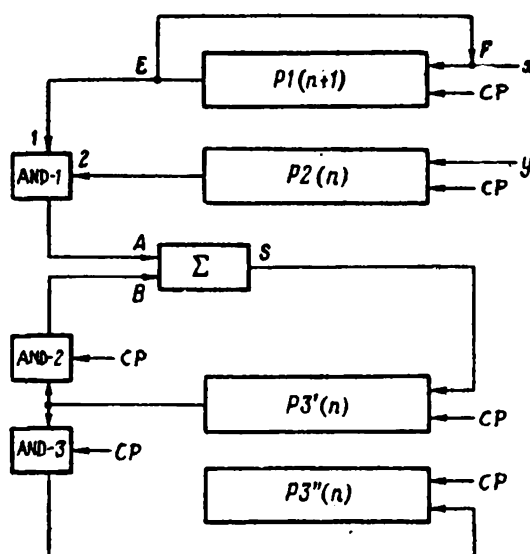


Fig.169 Coincidence-Type Multiplier with Serial Addition of Partial Products, Operating on Abbreviated Cycle

AND circuits.

Figure 169 shows the modified circuit of this unit. The multiplicand register P1 now has  $n + 1$  columns, the multiplier register P2 has  $n$  columns, and the product register is divided into two subregisters: P3' - the subregister of the  $n$  highest columns of the product, and P3'' - the subregister of the  $n$  lowest columns of the product. The AND-1 gate is used to form the partial products, and the AND-2 and AND-3 gates to distribute the digits of the product among the subregisters P3' and P3''.

In the initial state of the unit, the multiplier occupies  $n$ -columns of a

register P1, beginning with the extreme first. The multiplier fills the entire register P2, while the subregisters P3' and P3" are cleared. The partial products are formed and transferred to the input A of the adder in the same way as in the last multiplier. The only difference is that the value of each column of the multiplier is fed  $n + 1$  times instead of  $2n + 1$  times to the input of the AND-1 gate.

The formation of the full product by successive additions of partial products has the peculiarity that, after each addition, one more column of the sum takes the value of the corresponding column of the full product. For this reason, at each operation stage of the multiplier unit, a code pulse of the least significant column of the sum of the partial product can be propagated to the subregister P3" as one of the columns of the full product. In this case, after performance of all stages of the work of the unit, the  $n$  lowest columns of the full product will be recorded in the subregister P3".

In order to write the lower columns of the product into the subregister P3", the control pulses CP are fed to the AND-3 gate at a time corresponding to the first clock interval of the unit's operation at each stage, when the code pulse of the lowest column of the sum of the partial products appears at the output of the subregister P3'. The control pulses do not arrive at the AND-2 gate at these times. They are fed to it in such a way that the code /291 pulses of the columns of the sum of the partial products, beginning with the second, will pass to the input B of the adder.

The multiplication time for two  $n$ -digit binary numbers in this unit is

$$T_y = [(n + 1)n + 1] T.$$

This is a little more than half the time necessary to multiply two  $n$ -digit binary numbers in the unit shown in Fig.168.

Coincidence-type multiplication units with serial input of the digits of the partial products and parallel addition of the partial products. In such units, multiplication is by the principle of simultaneous formation of all partial products and their cascade-parallel addition to obtain the full product. This is one of the fastest practical multiplication units, despite the fact that they do operate with numbers represented in the form of serial pulse codes.

The cascade-parallel addition circuits in such units are so designed as to ensure simultaneous addition of all partial products pairwise, with gradual decrease in the number of summands to one, which is the full product.

Given eight partial products  $P_i$ , i.e., the full product

$$Z = P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7 + P_8.$$

the operation of cascade-parallel addition circuits can be illustrated by Table 21.

TABLE 21

## CASCADE-PARALLEL ADDITION OF PARTIAL PRODUCTS

Partial Products	1st Cascade of Adders	2nd Cascade of Adders	3rd Cascade of Adders
$P_1$ $P_2$	$P_1 + P_2 = \Sigma_1^1$	$\Sigma_1^1 + \Sigma_1^2 = \Sigma_2^1$	$\Sigma_2^1 + \Sigma_2^2 = \Sigma_3 = Z$
$P_3$ $P_4$	$P_3 + P_4 = \Sigma_1^2$		
$P_5$ $P_6$	$P_5 + P_6 = \Sigma_1^3$	$\Sigma_1^3 + \Sigma_1^4 = \Sigma_2^2$	
$P_7$ $P_8$	$P_7 + P_8 = \Sigma_1^4$		

The design and structural principles of a multiplier with cascade-parallel adders will now be considered on the example of a ferrite-transistor unit designed to multiply two fourteen-digit binary numbers. Figure 170 is the circuit of this unit. The unit consists of the multiplicand register P1, the multiplier register P2, the subroutine register P3, the groups of AND gates with two inputs AND<sub>1</sub> and AND<sub>2</sub>, four stages of coincidence adders with serial input of the columns of the summands  $\Sigma$ , an output AND<sub>3</sub> gate, and a generating cell GC to control the transfer of the product to other units. The simple ferrite-transistor cells are symbolized by rectangles on the diagram and the generating cells by rectangles with diagonal lines; each rectangle consists of two or three ferrite-transistor cells.

The multiplicand register P1 is an ordinary ferrite-transistor shift register with 15 columns. The two cells of the first column are an elementary delay line ensuring the simultaneous arrival of the serial code of the multiplicand and the codes of the digits of the multiplier at the AND<sub>2</sub> gate. From the outputs of the other 14 columns the serial codes of the multiplicand, mutually shifted a single clock, arrive at the inputs of the corresponding AND<sub>2</sub> gates.

The multiplier register P2 consists of 14 generating cells for the parallel transmission of the codes of all the digits of the multiplier to the AND<sub>2</sub> gates. The parallel input of the multiplier into register P2, i.e., the excitation of those generating cells to which the digit 1 in the particular column of the multiplier corresponds, is effected by the aid of AND<sub>1</sub> gates driven by pulses arriving from the first 13 columns of the subroutine register P3.

The multiplier is fed to the input of the unit in the form of a serial pulse code, the least significant columns first. When the code pulse of the first column of the multiplier arrives at the input of the unit, a pulse 1 is fed to the input of the subroutine register P3 from the control unit CU. This pulse enters the input of the AND<sub>1</sub> gate of the first column at the instant at which the code pulse of the first column of the multiplier arrives at its second



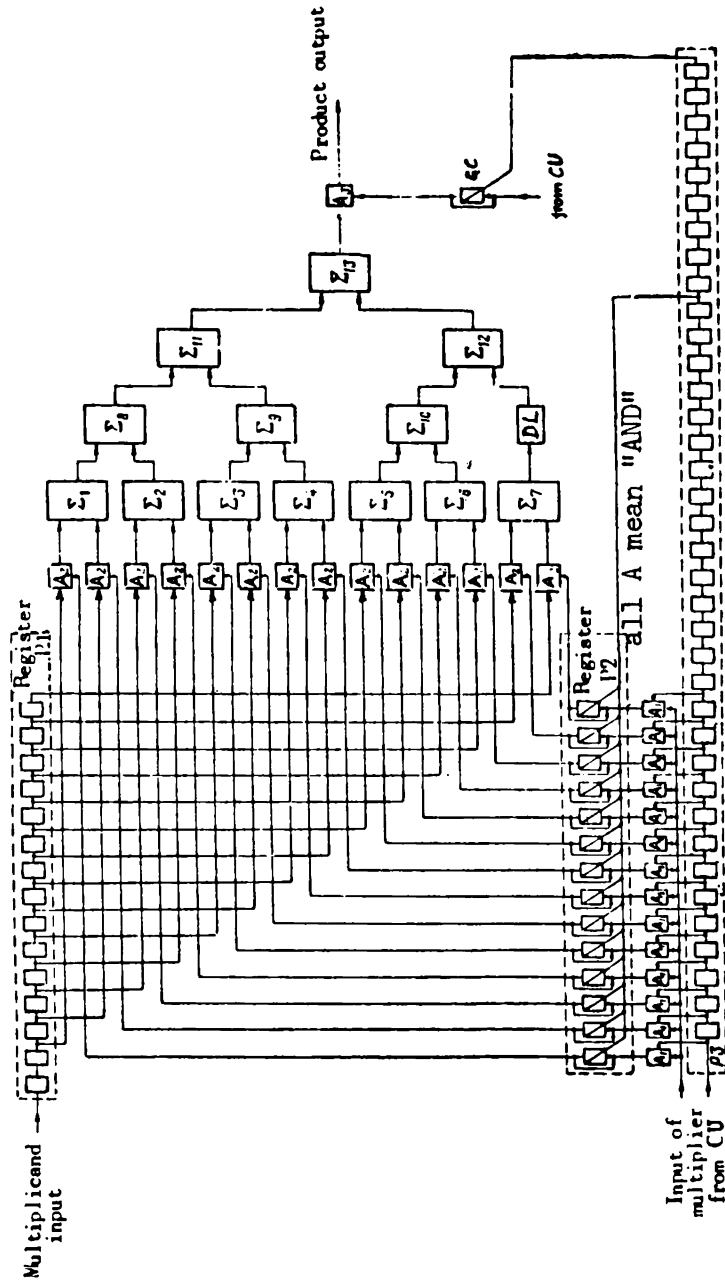


Fig.170 Multiplier Unit with Serial Input of Columns of Partial Products and Parallel Addition of the Partial Products

input. If there is a one in the first column of the multiplier, then a pulse that excites the generating cell of the first column of the register P2 is formed at the output of the  $AND_1$  gate. The control pulse is successively delayed by the columns of the register P3 by one clock each, so that it arrives at the input of the  $AND_1$  gate of the column to which corresponds the code pulse arriving at that time at the input of the unit. Thus, after the code pulse of the most significant column of the multiplier has appeared at the input of the unit, the generating cells of the register P2 will be excited in accordance with the parallel code of the multiplication unit.

The excited generating cells continuously send pulses of code 1 to the inputs of the corresponding  $AND_2$  gates; through these gates, the codes of the multiplicand, first shifted relative to each other in the register P1, arrive sequentially at the adders of the first stage. No pulses arrive at the inputs of the  $AND_2$  gates from the unexcited generating cells, so that these gates remain cut off at all times.

The partial products formed at the output of the  $AND_2$  gates are added (29) pairwise by seven serial adders in the first stage. The first pairwise sums of the partial products are added in the adders of the second stage, etc. Ultimately, the full product is formed as a serial pulse code at the output of the adder  $\Sigma_{13}$ .

Depending on the scales adopted in the computer, the product may be fed from the multiplication unit to other units, commencing at any desired column. For this purpose, a pulse from the control unit is fed at the required time to the generating cell GC controlling the output of the product. If the generating cell GC is unexcited, the  $AND_3$  gate is cut off and the product cannot pass from the output of the adder  $\Sigma_{13}$  to the output of the unit. If, however, a control pulse is fed to the generating cell in such a way that the first pulse appears at its output at the time when the code pulse of the 9th column of the product is formed at the output of the adder  $\Sigma_{13}$ , then this product will be fed to the other units starting with the 9th column.

The inhibit pulses that stop the generation of pulses by the excited cells are fed from the subroutine register P3. The generation can be inhibited only after formation of all partial products, i.e., after their passage through the  $AND_2$  gate. For this reason, the inhibit pulse is fed from the output of the 28th column of the register P3. The inhibit pulse can be fed to the generating cell driving the output of the product to other units only after the code pulse of the most significant column of the product has passed through the  $AND_3$  gate, as a result of which this inhibit pulse is taken off the output of the last - 37th - column of the register P3.

The multiplication time for two numbers in this unit is determined by the delay of the pulses in the register P1, the  $AND_1$  and  $AND_2$  gates, and the adders of all stages. For the 14-column unit designed on the system shown in Fig.170, this time  $T_y = 38T$ . If the working frequency of the unit is  $f = 100$  kc, then  $T_y = 380$   $\mu$ sec.

Coincidence-type multiplication units with parallel input of the columns

of the partial products and parallel addition of the partial products. Such units, like those with serial input of the digits from the partial products and parallel addition of those products, contain cascade-parallel addition circuits, except that each cascade or stage consists of a larger number of adders, owing to the partial input of the columns of the partial products.

Figure 171 shows the circuit of such a multiplier in which the cascade-parallel addition of the partial products is effected by means of the groups of OA-2 and OA-3 shown in Fig. 171. To simplify the circuit, the number of places of the co-factors is only four, i.e., the multiplicand  $X = x_4x_3x_2x_1$  and the multiplier  $Y = y_4y_3y_2y_1$ . The full product has eight places:  $XY = Z = z_8z_7z_6z_5z_4z_3z_2z_1$ .

The partial products are formed by means of four groups of AND gates, to whose second inputs the multiplicand is fed in parallel code. The multiplier arrives at the AND gates likewise in parallel code, the code pulse for its first column being fed to the first inputs of all the gates of the AND-1 group, the code pulse of the second column to the first inputs of the AND-2 group, and so on. As a result, the first partial product is formed at the outputs of the group of AND-1 gates, the second partial product at the outputs of the group of AND-2 gates, the third at the outputs of the group of AND-3 gates, and the fourth partial product at the outputs of the group of AND-4 gates.

The first cascade of the adders includes the adders  $\Sigma_1$  and  $\Sigma_2$ , the second the adder  $\Sigma_3$ . All these adders consist of OA-2 and OA-3 adders whose quantity is determined by the number of columns in the numbers. Thus  $\Sigma_1$  and  $\Sigma_2$  contain two OA-2 and two OA-3, while  $\Sigma_3$  contains two OA-2 and three OA-3. The mutual shift of the partial products during their addition is effected by fixed switching of the outputs of the AND gates, and of the inputs and outputs of the OA-2 and the OA-3.

The first and second partial products are added in the adder  $\Sigma_1$ , the third and fourth in the adder  $\Sigma_2$ . Their sums, in turn, are added in the adder  $\Sigma_3$ . As a result of all these operations, the full product in parallel code form is formed on the output buses of the unit. The code pulse of the most significant column of the full product appears at the output of the logic OR gate, which passes it either from the carry output of the OA-2 of the most significant column of the adder  $\Sigma_2$  or from the carry output of the OA-2 of the most significant column of the adder  $\Sigma_3$ .

A large amount of hardware is required to build these units. However, they are the fastest known; two numbers are multiplied in practically a single clock time.

The multiplication unit of the "Strela" computer is of this type.

Determining the sign of the product in multiplication units. In digital computers, as a rule, numbers are multiplied in the direct or sign-and-absolute-value code; the significant parts of the co-factors are propagated on separate code buses, and their sign digits on special buses.

Note: All A mean "AND"

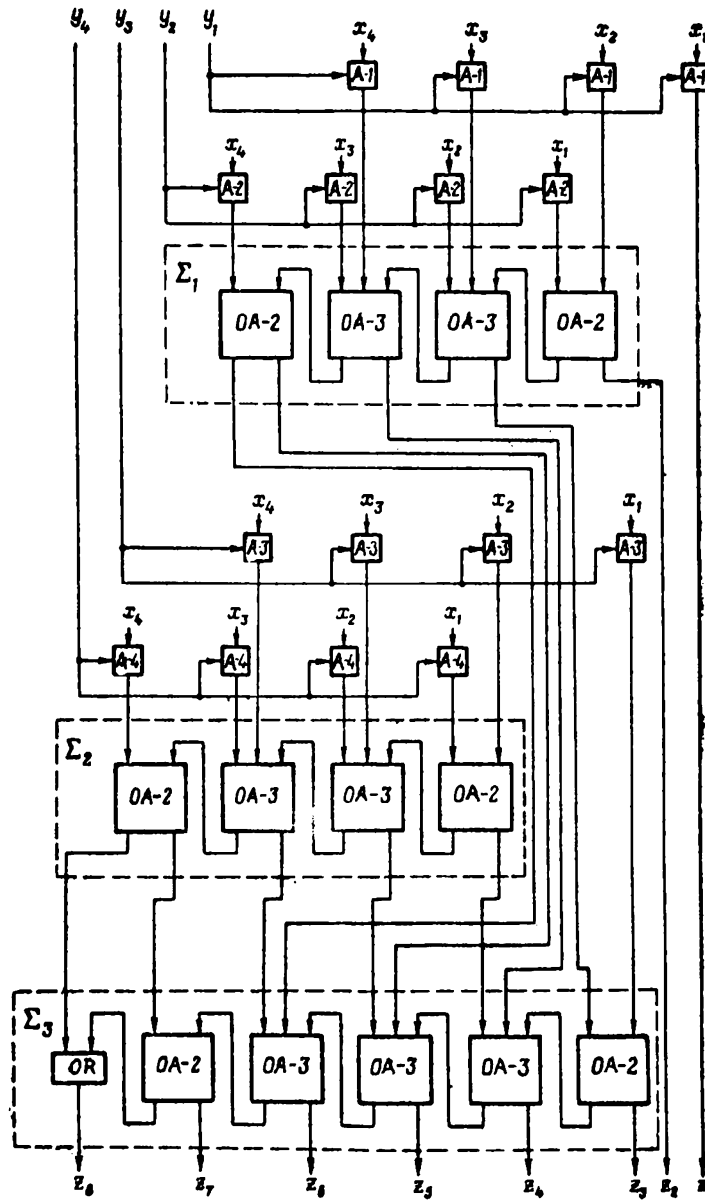


Fig.171 Multiplication Unit with Parallel Input of the Columns of the Partial Products and Parallel Addition of the Partial Products

According to the general rules of arithmetic, the sign of the product will be "+" if the co-factors have the same signs and "-" if they have different signs. In view of the generally accepted representation of signs, it is easy to see that their relations, for a given product sign, correspond to the relations of the input and output signals for an OR-OR gate. Such gates are therefore often used to determine the sign of the product. /297

A monostable flip-flop can also be used to determine the sign of the product. If the code pulses of the signs of the co-factors are fed successively to the counter input of such a flip-flop, this device will ultimately be switched to the position corresponding to the code of the product sign.

#### Section 44. Accumulators

Accumulators or counter-type adders are usually built from flip-flops and designed for the parallel input of the columns of the summands. The summands themselves, however, are introduced consecutively, first one summand, then the second. The accumulators most widely used are accumulating adders with serial carry and with simultaneous carry (or parallel carry).

Accumulator with serial carry. An accumulator with serial carry is the simplest adder with flip-flops and parallel input of the columns of the summands. It consists only of flip-flops and elementary delay lines, together with input and output gates.

Each flip-flop of an accumulator performs two main functions: adding the digits in a certain place and registration (storage) of the digit of the sum of that place. To perform these functions, all flip-flops of an adder must have combined (counter) inputs.

The number of flip-flops in an accumulator is determined by the number of columns in the summands. If an adder is designed to add the mantissas of numbers or of numbers with the point fixed in front of the most significant digit, then it contains  $n + 2$  flip-flops, where  $n$  is the number of digital places in the summands. Two flip-flops are allotted to take the sign of the sum, since the summands are usually represented in the form of modified codes.

Figure 172 is a schematic diagram of an accumulator-type adder with parallel input of the summands and serial carry.

The adder is designed to add numbers with  $n$  digits represented by a modified inverse code. The coupling circuit between the flip-flops contains the delay lines DL designed to delay the carry pulses, which is necessary in order to have the carry pulses arrive at the flip-flops for the more significant places only after all the transients due to the addition of the columns of the summands have been completed. A delay line is also connected to the end-around carry line, which couples the flip-flop for the most significant place of the sign TS 2 with the toggle for the lowest-order place  $T_1$ . /298

The summands are fed in inverse or ones-complement code nonsimultaneously

to the accumulator through the input gates  $B_0$ , unblocked by the control pulses  $CP_1$ . Since all the flip-flops of the adder were at first in the state 0, they will record the code of the first summand when it is fed. After the transients in the flip-flop have decayed, the second summand is fed in. The following

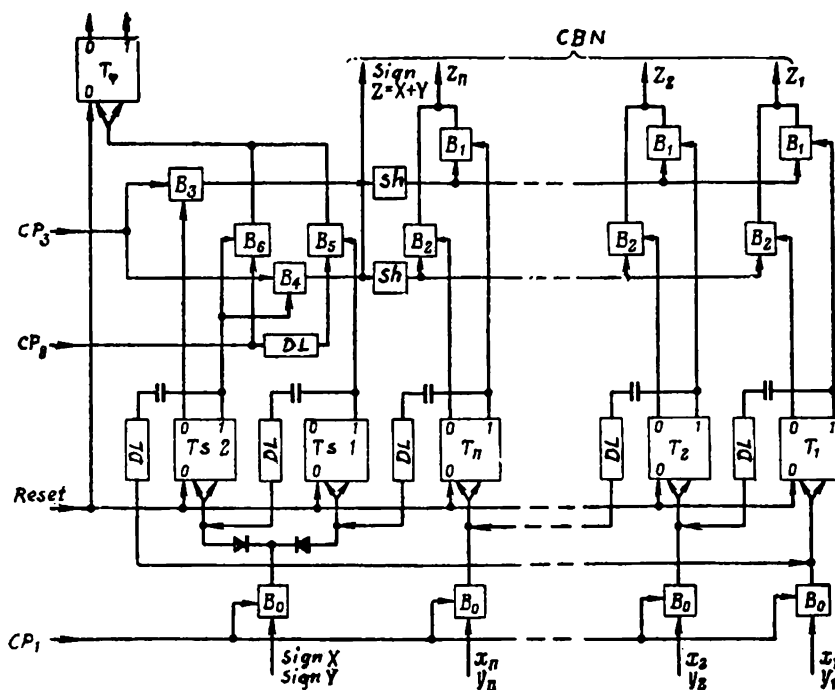


Fig.172 Accumulator with Parallel Input of Columns of  
Summands and Serial Carry

takes place now: If there is a zero in the given column of the first summand and a one in the second column, then the pulse arriving at the toggle input flips it from the state 0 to the state 1; but if there are ones in the given column of both summands, then the pulse flops the toggle from the state 1 to the state 0 and forms a carry pulse which arrives through the delay line at the input of the flip-flop of the next higher column. If the carry pulse is fed to the input of a flip-flop which, after adding the digits of the summands column, is in state 1, then it will reset this flip-flop to the state 0 and cause a further carry.

After all carries, the flip-flops of the adder are set to the states corresponding to the digits of the inverse code of the sum and will remain in these states as long as desired, so that the accumulator can also perform the functions of a sum register. /299

The shortest allowable latency time for the carry pulses on the delay line (DL) depends on the parameters of the toggles on which the adder is based. For most adders with vacuum or transistor toggles, this is a few microseconds. The maximum allowable latency time is determined by the required speed of the

accumulator. Since, with increasing delay time  $t_d$  of the carry pulses on the DL, the time required for adding numbers is also increased, usually a time  $t_d$  close to the minimum allowable is selected, namely, a few microseconds. Monostable multivibrators are used as the delay line to obtain such a delay time

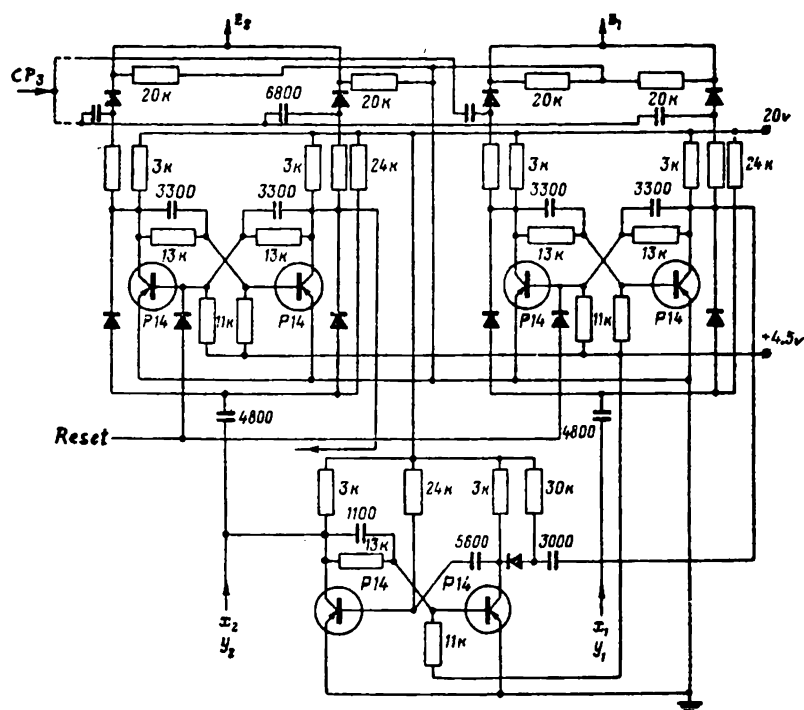


Fig.173 Schematic Diagram of Two Columns of an Accumulator

in some accumulators.

Figure 173 is a schematic diagram of two adjacent columns of an accumulator, using monostable multivibrators as delay lines. Here, each flip-flop and vibrator is based on two semiconducting triodes or transistors.

We have shown above that, in the accumulator under discussion, numbers represented by their ones-complement codes are added algebraically. Their sum will also be represented by ones-complement code, since end-around carry is accomplished in the accumulator. However, it is necessary to feed the value of the sum to other units in the direct or twos-complement code. Here, two groups of output gates  $B_1$  and  $B_2$  are connected to the accumulator. The gates  $B_1$  are connected to the right outputs of the flip-flops of the digital columns of their adder. These gates are opened when the corresponding toggles are in state 1. They are therefore called direct-code gates. The gates  $B_2$  are connected to the left outputs of the flip-flop. They are opened when the corresponding toggles are in state 0, and are therefore called inverse-code gates.

The transfer of the direct code of the sum to other units through the

number code buses (CBN) is controlled by the gates  $B_3$  and  $B_4$ , connected to the output of the flip-flop of the highest sign column TS 2. The gate  $B_3$  is opened if and only if the flip-flop TS 2 is in the state 0, and the gate  $B_4$  if and only if that flip-flop is in the state 1. The pulse  $CP_3$ , directly driving the transfer of the code of the sum from the adder, is fed simultaneously to both gates  $B_3$  and  $B_4$ . Passing through one of them, which is opened in accordance with the sign of the sum, this pulse arrives at the shaper Sh and then at the input of the gate  $B_1$  or  $B_2$ , so that the direct code of the sum is transferred to the other units.

The sign digit is fed with the values of the digital places of the sum to one of the code buses. If the sum is positive, then the pulse  $CP_3$  passes through the gate  $B_3$ , and nothing enters the sign code bus which corresponds to the code 0 or to the + sign. If the sum is negative, the pulse  $CP_3$  passes through the gate  $B_4$ , and thus the inverse code of the number formed in the accumulator, i.e., the inverse code of the direct code of the sum, is transferred. In this case, a pulse representing the code 1 or the minus sign, arrives at the sign code bus.

In adding fixed-point numbers, a check must be made for overflow after obtaining the sum. When numbers in modified codes are added, the indication for overflow is a difference in the sign places, and for this reason it is sufficient, in checking the correctness of the sum, to compare the sign digits.

In this system a separate flip-flop  $T_\phi$  whose state corresponds to the value of the criterion  $\phi$  is used to produce the criterion  $\phi$ , which is one in the case of overflow. To check the accuracy of the sum, the control pulse  $CP_2$  is applied. This pulse is propagated to the inputs of the gates  $B_5$  and  $B_6$  connected to the right outputs of the sign-digit flip-flops TS 1 and TS 2. If only one of these gates is open, which can happen only in case of overflow, /301 then a one pulse arrives at the input of the flip-flop  $T_\phi$ , flipping it to the state 1. However, if both gates  $B_5$  and  $B_6$  are cut off or opened, then the final state of the flip-flop  $T_\phi$  will be the zero state since, in the former case, no signal at all will arrive at its counter input and, in the latter case, two pulses will arrive at separate times, switching the flip-flop successively into states 1 and 0.

The following is the sequence of operation of the accumulator shown in Fig.172: First, a general clearing pulse is applied which sets the adder into the original position in which all the flip-flops are in the zero state. The inverse codes of the summands are then fed consecutively, while the control pulses  $CP_1$  are being fed to the gates. After input of the second summand, the inverse code of the sum is set up in the adder. To check the correctness of the sum, the pulse  $CP_2$  is applied under whose action the value of the criterion  $\phi$  is produced. If  $\phi = 1$ , the calculation is broken off; if  $\phi = 0$ , the control pulse  $CP_3$  is fed, which causes transmission of the direct code of the sum to the other units of the computer.

Since numbers are stored in direct code in the computer memory, an additional register is necessary to obtain their inverse codes before input to the accumulator. This additional register is connected with the adder through two



groups of gates having a function similar to the direct and inverse code gates of the accumulator proper.

In an accumulator with parallel input of the summand columns and serial carry, the maximum adding time  $T_{\max}$  of two  $n$ -digit binary numbers is determined primarily by the time required for the carry pulse to pass from the flip-flop of the lowest-order position to the flip-flop of the highest. Considering that the adding time is the time from application of the code pulses of the first summand to the adder to the time the flip-flops are set in the states corresponding to the digits of the sum,  $T_{\max}$  can be calculated by means of the formula

$$T_{\max} = T + (n + 2)(t_{tr} + t_d) + t_{tr}$$

where  $T$  is the time from application of the code pulses of the first summand to the adder to the time of application of the code pulses of the second summand;

$t_{tr}$  is the time taken by the transients in the flip-flop;

$t_d$  is the pulse delay time in the delay line.

In practical calculations  $t_{tr}$  is often neglected, and  $t_d$  is considered equal to  $T$ . Then,

$$T_{\max} = (n + 3)T.$$

If an accumulator is used in a multiplying system, only  $n$  of its flip-flops will be working, since the partial products are added without considering the signs of the co-factors. In this case a carry from the  $n$ -th digital place is possible, so that the maximum adding time  $T_{\max}$  is determined by the formula /302

$$T_{\max} = (n + 1)T.$$

Accumulator with parallel carry. An accumulating adder with parallel carry is not particularly fast, since the adding time for two numbers is proportional to the number of places they contain. Additional circuits are introduced for parallel (simultaneous) carry, which considerably shortens the adding time in these accumulators.

The principle of parallel carry is that an arriving carry pulse, when adding the digits of any column of the summands, is propagated in the direction of the higher columns while avoiding all flip-flops that are in state 1. If there is any flip-flop in state 0 on the path of such a carry pulse, then this pulse sets it in state 1 and is stopped from going beyond that point. All flip-flops that had been bypassed by the carry pulse are automatically reset to the state 0. This makes serial (cascade) carries superfluous.

The principle of parallel carry is illustrated by the example in Table 22 where, for comparison, we also give an example of serial carry.

TABLE 22

## ADDITION WITH SERIAL AND PARALLEL CARRY

Forms of Carry	Sign of Operation	Columns						Summands and Carries
		VI	V	IV	III	II	I	
Serial	+	1	0	0	1	1	1	Second summand
		0	1	0	0	0	1	First summand
	+	1	1	0	1	1	0	First intermediate sum
						1	1	Carry
	+	1	1	0	1	0	0	Second intermediate sum
					1	1	1	Carry
Parallel	+	1	1	0	0	0	0	Third intermediate sum
				1	1	1	1	Carry
	+	1	1	1	0	0	0	Sum

Accumulators use loops with gates, connected to one circuit or the other, for parallel carry. The system of connecting the gates in series with a special carry loop is most widely used. /303

Figure 174 shows the diagram of several places of an accumulating adder to illustrate the principle of the parallel carry loop. The gates of group  $B_1$  constitute the carry loop; they are directly driven by the flip-flops of the adder: If the flip-flop of a certain column is in state 1, then the corresponding gate is open and vice versa. The carry pulses produced at the flip-flop outputs pass to the carry loop through the gates of group  $B_2$ , which are driven by the clock pulses ClP fed from the control unit. A pulse arriving at the carry loop first passes through the gates of group  $B_1$ , corresponding to a flip-flop in state 0. In this case, the carry pulse passes through the delay line DL to the input of the flip-flop in state 0, and also to the inputs of /30

those flip-flops that were in state 1 and which had been bypassed by this pulse on the carry loop.

The carry pulses must be somewhat delayed in the gates of group  $B_2$  so that the pulse will arrive in the carry loop after the flip-flop of the next higher

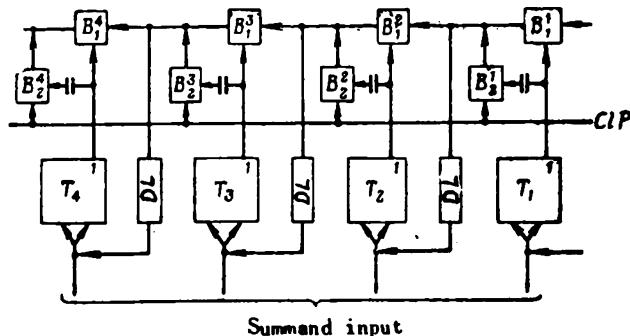


Fig.174 Build-Up of Parallel Carry Loops in an Accumulator

column has been set to state 1 or 0 by the code pulse from the second summand.

Assume that the number 0111 is written in the flip-flops of the adder, and let it be required to add the number 0001 to it. In this case, the initial state of the adder will be: flip-flops  $T_1$ ,  $T_2$ , and  $T_3$  in state 1, i.e., gates  $B_1^1$ ,  $B_1^2$  and  $B_1^3$  open; flip-flop  $T_4$  in state 0, i.e., gate  $B_1^4$  cut off. The number 0001 is fed to the toggles of the accumulator in parallel pulse code. The code pulse for the first place of this number flops the flip-flop  $T_1$  into state 0, causing the carry pulse to be produced at its output.

Simultaneously with the input of the code of the second summand into the adder, a clock pulse ClP is fed to the gates of group  $B_2$ . For this reason, the carry pulse arriving at the output of the flip-flop  $T_1$  passes through the gate  $B_2^1$  into the carry loop, along which it proceeds to gate  $B_1^4$  and, through the delay line DL, arrives at the inputs of the flip-flops  $T_2$ ,  $T_3$ , and  $T_4$ . The flip-flop  $T_4$  is set to state 1, and the flip-flops  $T_2$  and  $T_3$  to state 0. Pulses of the second carry thus are formed at the outputs of the flip-flops  $T_2$  and  $T_3$ . They cannot, however, reach the carry loop since the clock pulse opening the gates  $B_2$  is fed, in the case of addition of numbers, only when the code of the second summand is fed, and actuates the gates  $B_2$  at the instant when the formation of second carry pulses stops.

If, during the input of the code of the next summand, carry pulses are produced at the outputs of several flip-flops, then the propagation of pulses along the carry loops and the switching of the flip-flops to the states corresponding to the code of the sum proceed similarly to the processes described in the above example. Thus, the time required for adding two numbers in an accumulator with parallel carry loops is shortened to two working clocks, i.e.,  $T_{\Sigma} = 2T$ , and no longer depends on the number of columns in the summands.

## Section 45. Circuits for Operations of Multiplication and Division with Accumulators

Coincidence arithmetic units as a rule have no special multiplication or division equipment. When multiplication and division operations are being performed, the registers and the adder of such arithmetic units are switched by specific systems which, in most cases, are designed to utilize accumulators capable not only of adding numbers, but also of storing the intermediate and final results of the operations.

Multiplication circuits can be designed according to one of four possible variants. In practice, however, the only multiplication systems in wide use are those that shift the sums of the partial products in the adder and those that shift the multiplicand in the corresponding register of the AU. Of /305

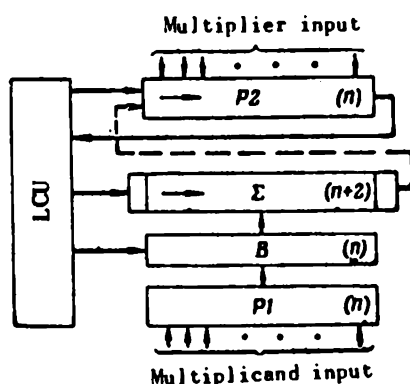


Fig.175 Scheme for Performing the Operation of Multiplication with Shifts of the Sums of the Partial Products in the Adder

the possible division schemes, the system shifting the dividend and remainders in the adder is most frequently employed.

When using a system that shifts the sums of the partial products in an adder, multiplication begins with the lowest-order place of the multiplier. In this case, the sequence of the elementary operations in multiplying two numbers is similar to the conventional procedure for performing this operation by hand, i.e., on a sheet of paper. The difference is only that in the arithmetic unit each of the newly formed partial products is directly added to the accumulated sum of the preceding partial products. The relative mutual shift of the partial product is accomplished by shifting their sum before each addition of the next partial product.

The organization of the relations between the parts of a coincidence arithmetic unit in performing the operation of multiplication by a system shifting the sums of the partial products in the adder is illustrated in Fig.175. This diagram shows that the register P1 is used to take the multiplicand and register P2 the multiplier; the partial products are formed by means of the group of

gates B, while the sums of the partial products and the full product are formed in the accumulator  $\Sigma$ . The sequence of operations in multiplication is controlled by the local control unit LCU which, in the general case, may form part of the control unit of the computer.

The registers P1 and P2 are of the n-column type, if the mantissas of the number being multiplied each have n digital places. The signs of the co-factors are usually fed to a separate system for determining the sign of the product, and therefore the sign columns of the registers P1 and P2 are not considered in multiplication. The register P2 must necessarily have shift lines to the right, toward the lower-order places. This is necessary in order, during the process of multiplication, to feed the LCU successively with the digits of the multiplier used to form the partial product.

The adder used in this system of multiplication has n main columns and 306 two auxiliary columns. The lowest of the auxiliary columns is used for rounding off the n-digit product, and the highest to take the carry one from the n-th main column, which can be formed on adding the next partial product to the accumulated sum of the partial product. There is a rightward shift loop for shifting the sums of the partial products toward the lower-order columns.

Multiplication in the scheme shown in Fig.175 is performed by successive shift of the codes in the adder and register P2 and sequential addition of the code transmitted through the gates B to the codes already in the adder. On the first shift, the value of the lowest place of the multiplier is transferred from the register P2 to the local control unit. If the lowest place of the multiplier was 1, then the local control unit emits a signal to open the gates B through which the multiplicand enters the adder as the first partial product. But if the lowest place of the multiplier was 0, then the signal to open the gates is not produced. This corresponds to a zero first partial product. On the second shift, the value of the second place of the multiplier is transferred from the register P2 to the local control unit, and the first partial product is shifted one column to the right in the adder. Then the second partial product formed by the aid of the gates B is added to the first partial product, and so on.

After all shifts and additions have been performed, and a shift is accomplished in the adder after each addition, the value of the full product of  $n + 1$  columns is formed in the adder; the  $n - 1$  lower-order places of the product are lost during the operation. The loss of these lower places has practically no effect on the accuracy of the result, since in each shift in the adder the lower place of the product that does not participate in the following additions is lost. In a computer operating with n-digit binary numbers, the result of the operation of multiplication must likewise be n-place. For this reason, the  $(n + 1)$ -place product actually obtained is rounded off to the value of the n highest places. The rounding-off operation is performed as follows: If the lowest auxiliary column contains a 1, then 1 is added to the lowest of the main n-columns. If, however, the lowest auxiliary column has a 0, then this addition of 1 is not performed. With this rounding, the error of the n-place product does not exceed a half-one in the least significant place.

In some cases it is necessary to preserve all  $2n$  places of the product. To prevent the loss of the less significant digits of the product shifted out of the adder during the shifts, they may be transferred to the cleared higher-order columns of the multiplier register P2. The transfer loop for the lower-order places of the product to the multiplier register is shown by the broken line in Fig.175.

The principal advantage of this multiplication scheme is its simplicity and the high accuracy of the result. Indeed, if only  $n$ -column registers and

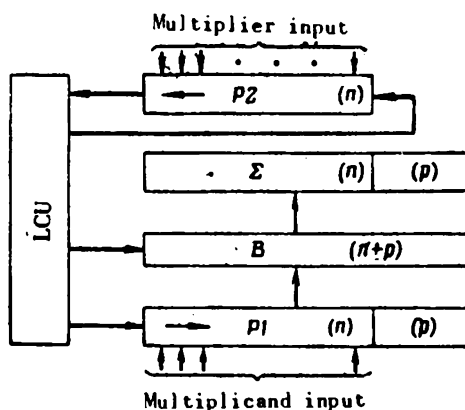


Fig.176 Multiplication Scheme with Shifts of Multiplicand in the Register

adders are used, it is possible to obtain the exact value of a  $2n$ -place product. If, however, the  $n$ -place product is formed, then to obtain it with an error not exceeding a half-one in the lowest-order place, it is sufficient to use only one auxiliary column in the adder (the sign column of the adder is ordinarily used as the highest auxiliary column). Another advantage of this scheme is the fact that it can be converted with relative simplicity into a division scheme.

Its disadvantages are the complication of the adder design by addition of the loops for shifting the code to the right, as well as the relatively long multiplication time. It is not possible to have the shift and adding period coincide in ordinary adders, and therefore for this system:

$$T_y = n(T_x + T_{sh}),$$

where  $T_{sh}$  is the code shift time in the adder for one place to the right.

The multiplication scheme in which the multiplicand is shifted in the register is shown in Fig.176. When this scheme is used, multiplication commences with the most significant digit of the multiplier, i.e., the partial product of greatest importance in the full product is formed first. The multiplicand, as before, is placed in the register P1, the multiplier in the register P2, and

the sum of the partial products formed by means of the group of gates B is accumulated in the accumulator  $\Sigma$ .

Since the multiplication begins with the most significant digit of the multiplier, the register P2 in this system has a leftward code shift loop. The adder itself has no shift loops, but to obtain the mutual shifts of the partial product, successive shifts of the multiplicand one place to the right in the register P1 are used. The absence of shift loops in the adder simplifies its circuit, which improves operating reliability in adding a partial product. This is an important advantage of multiplication with shifts of the multiplicand. /308

However, to perform multiplication by this system with the required accuracy, the multiplicand register, the adder and the series of gates need more hardware than in the system of Fig.175. If these building blocks are of the n-column type, then with each shift of the multiplicand during the operation its digits will be successively lost and the formation of the full product will take place by adding the "abbreviated" partial products. In this case the full product contains a large error. In the worst case, when the code factors have ones in all columns, the error of the n-place product is n - 1 units of its least significant digit.

To preserve all the places of the multiplicand during the shifts and obtain the accurate value of the product in this system, a 2n-column register P1 and adder may be used. This solution almost doubles the equipment required for the arithmetic unit and is obviously unrational. In practice, it is sufficient to obtain an n-place product with an error not exceeding the value of half a unit (or even a whole unit) of the least significant digit. It is therefore possible to use (n + p)-column multiplicand register and adder instead of the 2n-column type, the value of p being much less than n.

The number p of auxiliary columns of the multiplicand register and adder for the scheme shown in Fig.176 is calculated by the following simple formula:

$$2^{p-1} \geq n - p - 1.$$

The "Ural-1" computer uses this multiplication scheme, n = 35, p = 6, which is entirely in agreement with the results of the above formula.

Multiplication is performed in the scheme shown in Fig.176 by successive shifts and additions, as in the scheme of Fig.175. On the first shift, the multiplicand is shifted one place to the right, and the multiplier one place to the left. According to the value of the digit of the most significant place of the multiplier, the multiplicand is either transferred or not transferred to the adder as the first partial product. The multiplicand and multiplier are again shifted, etc. After n shift-and-add cycles, an (n + p)-place product is formed in the adder. The n main places of this product are rounded off to the value of the highest of the p additional places.

Since no shifts are performed in the adder itself, the addition cycles

coincide with the code-shift cycles in the multiplicand and multiplier registers. This coincidence decreases the multiplication time, determined by the formula

$$T_y = nT_z, \quad /309$$

if the clock time for addition by the scheme in Fig.176 is longer or equal to the clock time of the shift.

In addition to these variants of multiplication schemes, a scheme without shifting the code in the adder and registers is used in some coincidence and

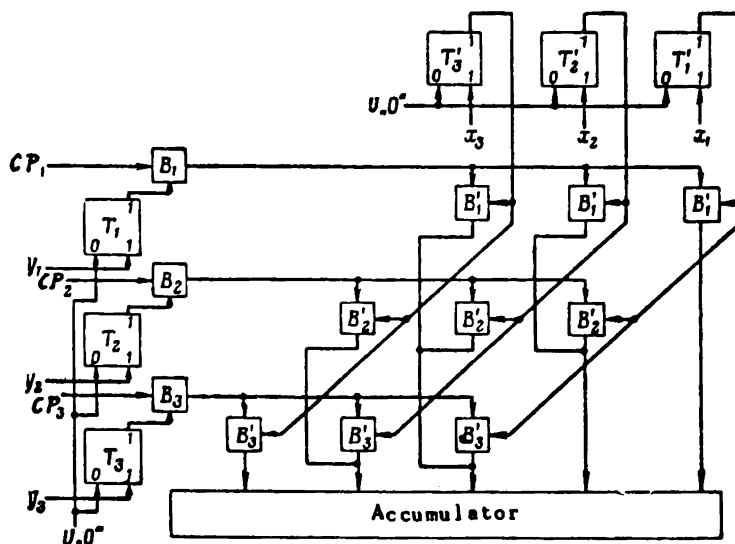


Fig.177 Multiplication Scheme with Registers and Adder without a Shift Loop

block arithmetic units. In such a scheme, the partial products are formed and mutually shifted by means of a matrix of gate circuits whose elements are subject to fixed switching among themselves and whose outputs are permanently connected to the inputs of certain columns of the adder.

Consider the design principle of the multiplication scheme without shifting in the adder and registers, on the example of the unit shown in Fig.177, designed to multiply three-digit binary numbers. The multiplier register consists of the flip-flops  $T_1$ ,  $T_2$ , and  $T_3$ , and the multiplicand register of the flip-flops  $T'_1$ ,  $T'_2$ , and  $T'_3$ . The gates  $B_1$ ,  $B_2$ , and  $B_3$  are used to obtain the signals of partial-product formation while the group of gates  $B'_1$ ,  $B'_2$ , and  $B'_3$  of the common gate matrix are used to form and mutually shift the partial products transmitted in parallel code to the accumulator.

The gates  $B_1$ ,  $B_2$ , and  $B_3$  are driven by the flip-flops  $T_1$ ,  $T_2$ , and  $T_3$ , respectively. Each toggle of the multiplicand register drives one gate of each group  $B'_1$ ,  $B'_2$ , and  $B'_3$ . Therefore, when a pulse is applied simultaneously to



the inputs of all the gates of any group, the parallel code of the multiplicand is formed at their outputs as a partial product.

The formation and transfer of the partial products to the adder is accomplished by feeding the control pulses  $CP$  to the gates  $B_1$ ,  $B_2$ , and  $B_3$ . Let  $101$  be the multiplier. Then the control pulse  $CP_1$  fed to the gates passes through gate  $B_1$  and arrives at the inputs of all gates of group  $B'_1$ , forming at their outputs the parallel code of the first partial product, which is written into the rightmost columns of the adder.

The second partial product is formed and propagated to the adder when the control pulse  $CP_2$  is applied. In this case, the pulse  $CP_2$  does not pass through the gate  $B_2$ , since  $B_2$  is cut off. Thus, nothing is transferred to the adder, which corresponds to zero value of the second partial product.

The third control pulse  $CP_3$  passes through the gate  $B_3$  and forms the third partial product at the outputs of the gates of group  $B'_3$ , which partial product is transferred to the adder with a shift of two places relative to the first partial product, on account of the corresponding fixed switching of the outputs of the gates of all groups.

The value of the product of the numbers being multiplied is set up in the accumulator after decay of the transients produced with the addition of the third partial product.

Realization of this scheme requires construction of a rather bulky matrix of gate circuits of which  $n^2$  are needed for  $2n$ -place products. The gate matrix can be simplified by using ferrite-core circuits. If the product is required to  $n$ -places with an accuracy to within one unit of the least significant place, then some of the least significant columns of the adder and the corresponding gates of the matrix can be eliminated entirely. In this case, it is sufficient to have an  $(n + p)$ -column adder, where the number of auxiliary columns of the adder  $p$  is calculated as for the scheme given in Fig.176.

Division schemes are usually formed by converting the corresponding multiplication schemes; if necessary, additional groups of elements are added to the overall scheme. Thus the scheme of Fig.175 can be converted into a division scheme by introducing in the adder and register  $P2$  leftward shift loops, and also by using the second group of gates to transfer the divisor to the adder in the inverse (or complement) code.

For a widely used division scheme with shifts of the divisor and re-/311  
mainder in the adder, an algorithm with restoration of the remainder has been adopted. This algorithm is formulated as follows for cases of division of the mantissas of numbers minus one:

- 1) Subtract the divisor from the dividend or compare them; if the dividend is less than the divisor, i.e., if the difference is negative, then division is possible since no overflow will occur; in the opposite case, a  $\phi$  signal is produced, on which the entire computer may stop.

2) Restore the dividend if, as a result of the first operation, a negative number is obtained, by adding the divisor to that negative number; double the dividend by shifting it one place to the left.

3) Subtract the divisor from the doubled dividend; if the difference is positive, then the first (most significant) digit of the quotient is 1, and the first remainder will be the difference between the dividend and the divisor; but if the difference is negative, then the first (most significant) digit of the quotient is 0, and the first remainder is the restored doubled dividend.

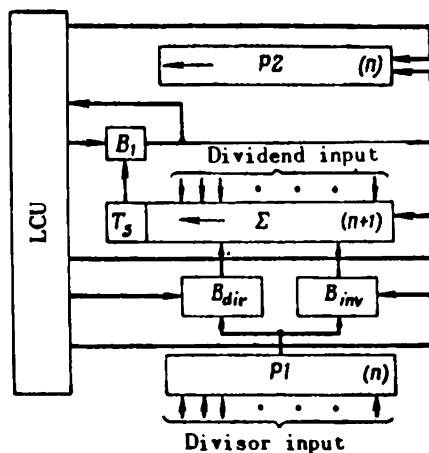


Fig.178 Scheme for Operation of Division

4) Double the first remainder by shifting it one place to the left.

5) Subtract the divisor from the doubled first remainder; form the second digit of the quotient and the second remainder by rules similar to (3), etc.

Layout of the links between the building blocks of a coincidence arithmetic unit in performing the operation of division by means of the above algorithm will be illustrated by the help of the diagram in Fig.178. This scheme shows that the register P1 is used to take the divisor, the register P2 to take the resultant digits of the quotient, and the accumulator  $\Sigma$  to take and shift the dividend and remainder. The gates of group  $B_{1,2}$  are used to transfer the divisor to the adder in inverse code, and the gates  $B_{dir}$  to transmit the divisor to the adder in direct code. The digits of the quotient are directly determined by means of the gate  $B_1$ . The local control unit LCU ensures the required sequence of the elementary operations during the overall operation of division.

The registers P1 and P2 have n-columns. The signs of the dividend and divisor are usually fed to a separate system for determining the sign of the quotient (this is usually coincident with the system for determining the sign of the product), for which reason the sign columns of the registers P1 and P2 are not considered in division. /312

The register P2, which takes the arriving digits of the quotient, the

higher places last, has a left-shift loop.

The adder used in this scheme has  $n + 1$  digit columns. The number of columns of the adder must be one more than that of the registers, so as not to lose the most significant digits on doubling the dividend and remainders. The sign column of the adder serves to determine the sign of the difference between the dividend (remainder) and the divisor. The flip-flop TS for this column, drives the gate  $B_1$ . When the flip-flop is in state 0, which corresponds to a positive difference between the dividend (or remainder) and the divisor, the gate  $B_1$  is opened; in the opposite case the gate  $B_1$  is cut off. It must be noted that, for correct formation of the sign of the difference, a code signal 1 is transmitted to the TS when the digital portion of the divisor is transmitted in inverse code to the adder.

Division by the scheme of Fig.178 is performed in the following sequence of operations: The possibility of division is first determined. For this purpose, the divisor is fed to the adder in ones-complement code. If the difference obtained is negative, then division is possible, and for this the dividend is restored by transmitting the divisor in sign-magnitude form to the adder. The dividend is then shifted one place to the left, and the formation of the digits of the quotient begins. The dividend is again fed in ones-complement code to the adder. After completion of the transient processes, a "query" signal is fed from the local control to the gate  $B_1$ . If the remainder is positive, then gate  $B_1$  is opened and the value of a one of the most significant place of the quotient is recorded in the register P2. In the opposite case, the gate  $B_1$  is closed, and the value 0 is recorded in the register P2 for the most significant digit of the quotient. Then the figures in the register P2 and adder are shifted one place to the left, a negative remainder in the adder having first been restored, the divisor being fed to the adder in inverse code, etc.

After  $n$  such successive steps, the value of the quotient is set up in the register P2. Its most significant digit, after all these shifts, is now in the leftmost column. Obviously, for each step of division a single shift and a single addition must be performed; if the remainder has to be restored, the number of additions per step increases to two. Thus, the average division time in the scheme of Fig.178 is determined by the formula

$$T_{div}^{av} = n \left( \frac{3T_v}{2} + T_{sh} \right)$$

The division time can be somewhat shortened if division is performed by an algorithm without restoring the remainder, as is done in the Ural-2 computer. However, this involves some decrease in accuracy.

## Section 46. Coincidence-Type Arithmetic Units

Arithmetic units of the coincidence (universal) type are used in most modern general-purpose and special-purpose machines. Such arithmetic units

usually consist of flip-flop registers, an adder, groups of gates, and certain other auxiliary pieces of hardware.

Coincidence-type arithmetic units, based on flip-flop circuits, can be subdivided into two principal forms:

Units with accumulators and toggle registers (AU with explicit adders); the BESM-2, Ural-1, Ural-2, Ural-4 and other computers have units of this type.

Units with toggle registers and groups of elements to perform the elementary logic operations (AU without explicit adders); the M-2, Minsk-1 and other computers have units of this type.

An arithmetic unit with an explicit adder is based in most cases on an adder with parallel input of the columns of the summands and with parallel carry. In addition, the AU includes two or three flip-flop registers, groups of gates, and a local control unit. The registers, and sometimes the adder, have several code shift loops.

The adder is designed to add the number codes, while the registers receive, store, and shift the codes of the numbers on which the various operations are performed and, in conjunction with the groups of gates, convert the sign-magnitude form of a number into its two- or ones complement and vice versa. The results of the operations are fed to the other units either from the adder or from one of the registers, depending on the type of operation performed.

Arithmetic units with explicit adders are designed to perform several elementary operations that form part of any arithmetic or logic operations to be performed in the computer. These operations, for example, may include:

Receiving, in the functional blocks of the arithmetic unit, the code of the number from the memory unit or some other unit of the computer.

Transmitting the code of a number to the memory unit or some other unit of the arithmetic unit.

Converting the direct code of a number into the complement (or inverse) code, or vice versa.

Shifting the code of a number toward the more significant or less significant places.

Performing addition of number codes.

/31

Resetting the flip-flops of the registers and adder to the state 0.

The required arithmetic or logic operation is performed in the unit under the action of signals arriving from the local control and requesting the building blocks of the AU to perform elementary operations in a certain order. The adder of the AU plays the major role here.

An arithmetic unit without explicit adder is based on four flip-flop reg-

isters, in the coupling loop between whose columns the AND, OR, and OR-OR gates are connected.

The flip-flop registers are used to receive, store, and shift the codes of the numbers on which arithmetic and logic operations are being performed, to

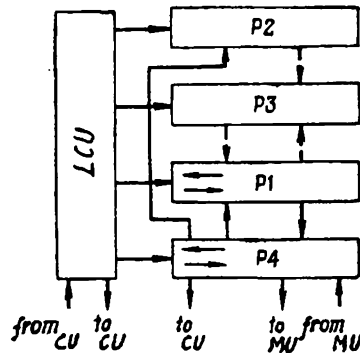


Fig.179 Block Diagram of Four-Register Arithmetic Unit

form the intermediate and final results of the calculations, and to feed these to the other units of the computer. The AND, OR, and OR-OR gates are used to make sure that the operations on the numbers in this arithmetic unit are performed according to the rules.

Figure 179 is a general block diagram of a four-register AU without explicit adder. The unit is connected with the memory unit through the register P4, which directly receives the numbers from the MU and feeds the results into it. The criteria of the results are also transmitted from this register to the control unit CU.

The codes of the numbers can be fed from the register P4 to the registers P1 and P2, and the results of the operations, formed in register P1, to the register P4. The links between the registers through the groups of logic circuits, modifying the codes fed to their inputs, are shown on the diagram by broken lines.

The local control LCU is used to control the operation of the arithmetic unit during its operations on numbers. It produces certain series of instruction pulses which are fed to the registers and logic circuits of the arithmetic unit under the action of signals arriving from the central control unit CU.

For certain operations there are shift loops in the registers P1 and P4 (shown on the diagram by reversed arrows).

Arithmetic units of this type are designed to perform the following arithmetic and logic operations: addition, subtraction, multiplication, division 315 and comparison, and sometimes also for logical multiplication (separating part

of a number). Addition is the most important of these operations, all of the other operations ultimately reducing to it.

Addition is performed by special rules in two steps.

The first step is to determine and record the binary carry ones in all columns. The binary carry one in the  $i$ -th column is recorded in the following cases:

if the  $(i - 1)$ -th columns of the summands contain ones;  
 if the digits in the  $(i - 1)$ -th columns of the summands are not the same and there is a carry from the  $(i - 2)$ -th column to the  $(i - 1)$ -th column.

TABLE 23

FORMATION OF BINARY CARRY ONES AND OF THE SUM

Columns	VI	V	IV	III	II	I
1st summand . . .	1	0	0	1	1	0
2nd summand . . .	0	0	1	0	1	1
Codes of carries . .	0	1	1	1	0	0
Value of sum . . .	1	1	0	0	0	1

The second step is to determine the magnitude of the sum by changing the digits of the first summand according to the following rules:

The digit of a given place of the sum equals the column of the first summand, if the carry to that column is the same as the column of the second summand.

The digit of a given place of the sum is different from the column of the first summand, if the value of the carry to that column is not the same as the value of the column of the second summand.

The formation of binary carry ones and of the sum of two numbers by these rules will be illustrated by the example given in Table 23.

The rules for performing the operation of addition in two steps may be written in the form of logic functions. For this purpose, we introduce the following notation:

$P_i$  = signal of carry to the  $i$ -th column, corresponding to 1;  
 $P_{i-1}$  = signal of carry to the  $(i - 1)$ -th column, corresponding to 1;  
 $x_i(\bar{x}_i)$ ;  $x_{i-1}(\bar{x}_{i-1})$  = digits of the  $i$ -th and  $(i - 1)$ -th columns of the

first summand, corresponding to 1 and 0;

$y_i(\bar{y}_i)$ ;  $y_{i-1}(\bar{y}_{i-1})$  = digits of the  $i$ -th and  $(i - 1)$ -th columns of the second summand corresponding to 1 and 0;

$S_{x,i}$  = signal for change in the value of the i-th column of the first summand in forming the sum.

It follows directly from the rules that

/316

$$P_i = x_{i-1}y_{i-1} + (x_{i-1}y_{i-1} + \bar{x}_{i-1}y_{i-1})P_{i-1};$$

$$S_{x_i} = (P_i \bar{y}_i + \bar{P}_i y_i) (\overline{P_i y_i + \bar{P}_i \bar{y}_i}) = P_i \bar{y}_i + \bar{P}_i y_i.$$

Thus to form the binary carry ones, two AND gates with two inputs, one OR gate with two inputs, and one OR-OR gate must be used in each column. To form the signals for change in the values of the columns of the first summand,

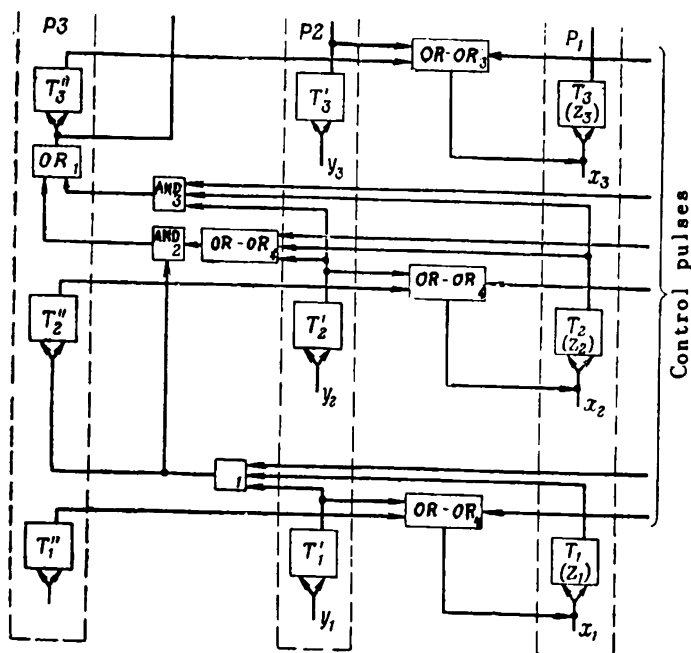


Fig.180 Structural Schemes for Addition in Arithmetic Units without Explicit Adder

a single OR-OR gate in this column is sufficient.

In performing the operation of addition, the first summand is placed in the register P1, the second in the register P2, and the binary carry ones are taken by the register P3. The sum is obtained in the register P1.

The structural scheme of the three least significant columns of the second part of the register, during their addition, including the loops for forming

the values of the carries and the final sum, is shown in Fig.180. The first summand  $X = \dots, x_3x_2x_1$ , the second  $Y = \dots, y_3y_2y_1$ , and the sum  $Z = \dots, z_3z_2z_1$ . The links between the columns of the registers required to form the values of the carries and the final sum, are constructed in strict accordance with the /317 above rules and logic functions. In these links, AND, OR, and OR-OR gates are used. When connected to the register flip-flops, they have three inputs each instead of two. The inputs are connected to the potential outputs of the flip-flops, and the control pulse from the local control element is fed to the third input. This pulse passes to the output of this scheme only if the signals of code 1 at the first two inputs agree for the AND gate and if the signals disagree for the OR-OR gate. It is natural to use such gates, since their main inputs are potential, while the output must be pulsed.

In the M-2 and M-3 computers such gates use semiconductor diodes, and the OR-OR gate is called the non-coincidence decoder.

The circuit forming the value of the carry in each column except the second includes two AND gates, one OR-OR gate, and one OR gate. This circuit consists of two branches, each corresponding to one of the two rules for formation of binary carry ones.

The branch of the carry loop to the third column, obeying the first rule, includes the  $AND_3$  and  $OR_1$  gates. Let us assume that the second columns of the summands are 1. Then the flip-flops  $T_2$  and  $T'_2$  will be in the state 1 and open the  $AND_3$  gate. The control pulse will pass through the  $AND_3$  gate and then through the  $OR_1$  gate, and will ultimately set the flip-flop  $T'_3$  to state 1.

The second branch of this loop, including the  $AND_2$ ,  $OR-OR_4$ , and  $OR_1$  gates, obeys the second rule of formation of the binary carry ones. In fact, if the digits in the second columns of the summands are different, then the flip-flops  $T_2$  and  $T'_2$  will be in different states and will thus open the  $OR-OR_4$  gate, which will form a signal that will open the  $AND_2$  gate. Now, in the presence of a carry pulse from the first column to the second, this pulse can pass through the  $AND_2$  gate and then through the  $OR_1$  gate, ultimately setting the flip-flop  $T'_3$  to state 1.

The loop forming the value of the sum in each column includes one OR-OR gate. The gate in the loop for forming the sum in the third column is called  $OR-OR_3$ . It fully ensures satisfaction of the rules of sum formation. In fact, if the value of the carry to the third column is the same as the digit of the third column of the second summand, then the flip-flops  $T''_3$  and  $T'_3$  will be in the same states, so that the  $OR-OR_3$  gate will be cut off. The control pulse cannot pass through it and arrives at the input of the flip-flop  $T_3$ , which remains in the previous state, showing the digit of the first summand as the digit of the sum.

If, however, the value of the carry to the third column is not the same as the digit of the third column in the second summand, then the flip-flops  $T''_3$  and  $T'_3$ , being in different states, will open the  $OR-OR_3$  gate. The control /318 pulse will pass through this gate and switch the flip-flop  $T_3$ , obeying the rule for the change in digit of the first summand in forming the sum.



When this arithmetic unit is used on a fixed-point computer, it is designed to conduct operations on the numbers in modulus form. The sign of the result is determined in a special building block based on a flip-flop. The complement of a number fed into the registers P1 and P2 can be formed there for the operations of subtraction and comparison.

An indication for overflow during addition is the formation of a carry 1 in the  $(n + 1)$ -th column where  $n$  is the maximum number of columns of the summands. The register P3 has a special flip-flop  $T_{\varphi}$  to indicate this carry 1.

The operation of comparison of two numbers is performed as follows: The numbers  $X$  and  $Y$  are placed in the registers P1 and P2, respectively. In the register P2 the complement of the number  $Y$  is taken:

$$Y' = 1 - |Y|.$$

Control pulses are then sent to the logic circuits of the carry loops to show the ones in the corresponding flip-flops of the registers P3. It is then easy to determine from the state of the flip-flop  $T_{\varphi}$  which of the numbers,  $X$  or  $Y$ , is greater. In fact, the sum of the numbers  $X$  and  $Y'$  is

$$S = |X| + |Y'| = 1 - (|Y| - |X|).$$

$$\text{at } |X| < |Y| S < 1,$$

$$\text{at } |X| > |Y| S > 1,$$

$$\text{at } |X| = |Y| S = 1.$$

This means that, if the flip-flop  $T_{\varphi}$  is in the state 1, then  $|X| \geq |Y|$ , but if it is in the state 0, then  $|X| < |Y|$ .

Thus, in comparing the absolute values of two numbers, no operation is performed on the numbers themselves, except for taking the complement of one of them, which does not result in its loss. This permits us to simplify the operation of division.

In subtraction, the minuend  $|X|$  is placed into the register P1, and the subtrahend  $|Y|$  into the register P2. The difference is formed in the register P1.

Subtraction in an arithmetic unit consists in calculating the value of  $|X| - |Y|$  if  $|X| > |Y|$ , or of  $|Y| - |X|$  if  $|Y| > |X|$ . The difference of two numbers is obtained by adding the minuend to the complement of the subtrahend, disregarding the carry to the  $(n + 1)$ -th column (for determining the absolute value of the difference).

The relation between the numbers  $|X|$  and  $|Y|$  is determined by the comparison, taking the complement of the number in the register P2. If the flip-flop  $T_{\varphi}$  is in the state 1 as a result of the comparison, then  $|X| > |Y|$ , and

the absolute value of the difference is formed by addition in the register P1. But if the flip-flop  $T_\varphi$  is in the state 0, then  $|X| < |Y|$ . In this case, the complements of both numbers  $|X|$  and  $|Y|$  must be taken in the registers P1 and P2, and addition must be performed only after this is done. The result of that addition will be the formation of the absolute value of the difference in the register P1.

In multiplication, the multiplicand is placed into the register P2 and the multiplier into the register P4. The sum of the partial products is then accumulated in the register P1.

Multiplication reduces to the alternating shifts of the contents of the registers P1 and P4 and adding the content of the register P1 to the multiplicand in the register P2. In this case, addition will take place if and only if the flip-flop of the least significant column of the register P4 is in the state 1 (the shifts are toward the less significant places).

In the operation of division the dividend is placed into the register P1 and the divisor into the register P2. The quotient is formed, column by column, beginning with the most significant column, in the register P4.

Division is performed in the following sequence, which is the same as the usual rules for division:

1. After placing the divisor into the register P2 and the dividend into the register P1, the complement of the divisor is taken and the carry one is placed in the register. If this flips the flip-flop  $T_\varphi$  to the state 1, then division is impossible since the dividend is greater than the divisor and the absolute value of the quotient will be greater than unity. But if  $T_\varphi$  is in the state 0, then division is possible and the arithmetic unit proceeds to perform it.

2. The division begins with the shift of the dividend and quotient one place to the left; this sets 0 in the flip-flop of the least significant column of the quotient register P4. If the flip-flop  $T_\varphi$  and the flip-flop of the  $(n + 1)$ -th column of the register P1 are then in different states, the sum will be formed on the next cycle in the register P1 (i.e., the first remainder will be formed) and the state of the flip-flop of the lowest-order column of the register P4 will change from 0 to 1. But if the flip-flop  $T_\varphi$  and the flip-flop of the  $(n + 1)$ -th column of the register P1 are set in the same state, then the sum will not be formed in the next cycle, and the flip-flop of the least significant column of the register P4 remains in the state 0.

3. The number of successive cycles, each consisting of one clock interval of a leftward shift by one column in the registers P1 and P4, and one clock interval of sum formation, equals the number of places in the numbers.

## INPUT AND OUTPUT UNITS

Section 47. Input and Output Devices in General

The data input and output units for both calculating or problem-solving computers, and control computers, are designed for the transition from one representation of the input data and problem-solving results to another representation.

In spite of the common purpose, the devices for problem-solving and control computers differ substantially. The reason is that, in computers of the former type, the input and output data, like the digital codes of the numbers with which the computer operates, are discrete quantities differing from the number codes only in the form of representation. In control computers, the difference is more profound, lying in the very form of representation: The machine operates with the digital codes of numbers which can be assigned only discretely, while the input and output data are represented for the most part by continuous quantities: shaft angles, voltages, etc.

The practice of building and using input and output devices shows that it is considerably simpler to pass from one type of representation of quantities to another than from one form of representation to the other, i.e., from continuous quantities to discrete, or from discrete quantities to continuous. In the former case, all that is necessary is to represent the initial numerical material in such a manner that it can be introduced into the computer with simultaneous translation of the numbers from one number system to another. Similarly, the conversion of the results of the solution generally consists merely in the conversion of numbers. The transition from continuous to discrete quantities, however, requires rather complicated methods of correlating discrete and continuous quantities, usually varying at high rates and over wide ranges.

There is still another important difference between the input and output devices for problem-solving and control computers. The input and output devices of the former type operate asynchronously with the other units of the computer, since they are slower. The character of problem-solving on such machines does not, in principle, require both input and output to be synchronous with the other units, since the input data for solving a single problem are fed to the computer in practically a single block and in good time, while the output data can be printed even after the problem-solving process in the computer has /321 been completed. In this respect, the input and output devices of such computers are sometimes called external units, i.e., units not directly connected with the main units of the computer (arithmetic and memory).

In control computers, the input and output devices operate synchronously with the other units of the machine, and renew at high speed the input data and output data (control signals) fed to the controlled object. Synchronous opera-

tion of all units of a control computer is particularly necessary when the machine is directly connected with the controlled object by electric or mechanical links.

The initial data and the program for problem solution are fed into problem-solving computers mainly by means of punched cards and punched tapes. The input units of such machines usually consist of a set of devices for punching cards or tapes, i.e., for encoding the original numerical material into a system of punches (holes), for checking the accuracy of punching, for reading the number codes from the punched tapes or punched cards, and for writing them into the memory of the machine. If necessary, the numbers are translated from one number system into another.

Magnetic recording is sometimes used for the input of data to such machines: the coded numerical material is copied from the punched cards or punched tapes onto magnetic tape and is fed to the computer by readout from this magnetic tape.

A problem-solving computer has a purely documentary connection with the input devices. The input data are transferred by the human operator from the input units to the computer in the form of documents. Such documents may be:

Punched cards with parallel row-by-row writing of the numerical data (Strela computer).

Punched tape with parallel-serial writing of the numerical data (Ural computer).

Punched tape with serial writing of the numerical data (BESM-2, M-2, and M-3 computers).

Magnetic tape with parallel row-by-row writing of the numerical data (Strela computer) or with parallel-serial writing (Ural computer).

Magnetic tape with serial writing of the numerical data (BESM-2 computer).

To prepare and check punched cards and punched tapes, slightly modified standard telegraph equipment and the equipment of counting- and-punching machines is ordinarily used. When punched cards and punched tapes of the types used in the Ural computer are employed, the system includes a keyboard for selecting and coding the original numerical data; punches which directly punch the cards (or tapes); control devices to check the accuracy of the entire <sup>1322</sup> system on the punched cards or punched tape; readers (transmitters) for reading the number codes from the punched cards or punched tapes and writing them in the appropriate registers or memory units of the computer.

If narrow punched tape - ordinary telegraph tape - is used, the input system includes punches and standard telegraph equipment and transmitters (for instance, the T-50 transmitter in the M-3 computer).

Almost all problem-solving computers are arranged for input of the numerical data to the memory unit directly from the control console, on which the keyboard is usually placed. In most of these computers, this input method is used only for correcting the data already introduced and for input of individual quantities. In some computers, however, this is the only input method.

The results of solutions in these computers are read out either by punching cards or tapes or by printing the numbers in separate Tables. Magnetic recording can also be used.

The output devices include punches, special printers, telegraph teletypes and photoprinting devices. Some non-control computers use other output devices. Thus, the M-3 has a special cathode-ray indicator to display the result of a solution in the form of a graph on its screen.

In control digital computers, the main input equipment is used to pass from continuous to discrete quantities. Most of them require conversion of continuously varying input shaft angles, or voltages, to the corresponding sequences of number codes in the number system used by the computer. This digit encoding of shaft position or of voltage is effected by the aid of "voltage-to-number" (voltage digitizer) and "shaft-to-number" (shaft digitizer) devices, i.e., analog-to-digital converters.

Besides these devices, the input system of control machines also contains equipment for the input of discrete information.

The output devices from control digital machines mainly takes care of the transitions from discrete quantities to continuous. These are devices of the "number-to-voltage" and "number-to-shaft" types, i.e., digital-to-analog converters.

#### Section 48. Input Devices for Problem-Solving Computers

/323

As indicated above, data input to such computers is primarily from punched tape and punched cards. The input units therefore consist mainly of devices for preparing the punched tape and cards, and devices for reading the data from these documents and effecting a direct input of the initial numerical data into the memory unit of the computer.

Punched cards and punched tapes are prepared in the same sequence.

Let us familiarize ourselves with the sets of input units for computers with punched cards and wide punched tape, using the Ural-1 computer as an example.

Figure 181 is a block diagram of the input system of this computer. The input devices include the keyboard unit KU, the checker reader CR, the input punch IP, and the card reader R.

The block diagram shows the principal links between the devices. The

heavy lines give the electric connections and the broken lines the documentary link, i.e., the link by means of punched tape transferred by the operator from one device to another.

The reader, in contrast to the other units, is directly connected with the

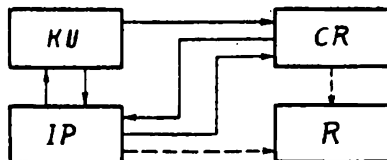


Fig.181 Block Diagram of Input Devices of the Ural-1 Computer

computer and operates together with the arithmetic and memory units.

Consider the design and operation of each of the input devices.

The keyboard is designed to convert automatically the original numerical material - the program of problem solution and the initial data - into the binary and binary-coded decimal systems and to produce the corresponding electric signals to be fed to the input punch for punching the tape. These functions are performed by the keyboard building block which is the principal part of the keyboard unit. The keyboard unit, besides this building block, contains a printing device for printing the selected numbers on paper.

The decimal and octal numbers constituting the initial data and the program of problem solution are directly set up on the keyboard. The keyboard of this keyboard unit is complete, and all digits of one of the decimal numbers by means of which the initial data are recorded, or of one of the octal numbers by means of which the program of the problem-solving is recorded, can be set up on it simultaneously. In the Ural-1 computer, the greatest number is expressed by nine decimal places. The keyboard is designed to take the same number of digits.

Figure 182 shows the keyboard of the keyboard unit. It consists of 324 nine digital sections, a sign section, and a control section. Each digital section has ten keys marked from 0 to 9 to set up the decimal digits. The extreme left digital section corresponds to the most significant place. To set up decimal numbers, the keys of all nine digital sections are depressed. For octal numbers, only the keys of the six left digital sections are depressed, since octal numbers in this system are usually of the six-place type.

The sign section has two keys, "-" or the sign of a negative number, and "Z" for the symbol of the zone which is punched on the tape.

The control section has three keys: skip or space "SK", operate "O", and return "R".

The printing assembly of the keyboard system is a printer controlled by signals from the keyboard. When the keys are depressed, the printer prints the appropriate characters on a special blank. The printing of the numerical material set up on the keyboard facilitates checking the selection of the numbers and punching of the tape.

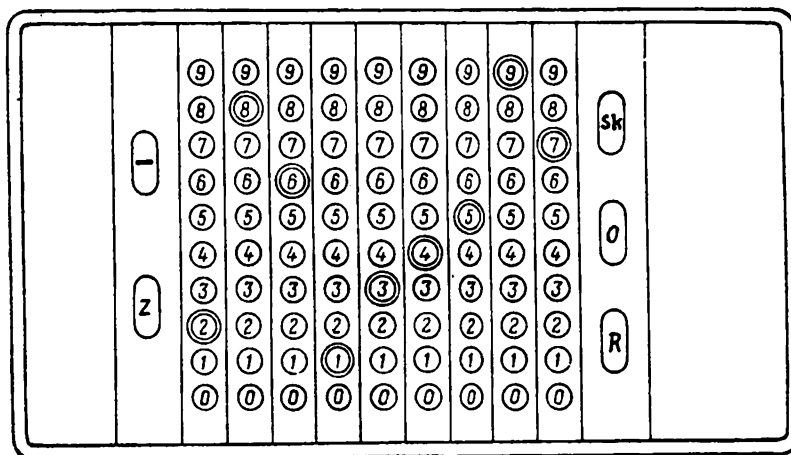


Fig.182 Keyboard of Keyboard Unit of the Ural-1 Computer

The electric circuit of the keyboard device includes decoders to translate numbers from the decimal system to the binary-coded decimal and from the octal to the binary, as well as control elements for the number selection and printing building block.

Each decoder consists of contacts of one digital section of the keyboard. The decoder is reset by depressing one key or the other, since certain contacts of the decoder are thus closed. /325

Figure 183 is a schematic diagram of the decoder of the keyboard device. The decoder has four inputs and four outputs. The output signals arrive by way of the blocking diodes  $D_1 - D_4$  in the common output circuit of the given place of the number. The signals (pulses)  $P_1 - P_4$  arrive successively at the inputs of the decoder from the input punch. Figure 183 shows the distribution of the signals among the decoder inputs and by time. The diagram indicates that first the signal  $P_1$  arrives at the first input (In. 1) then the signal  $P_2$  at the second input (In. 2), and so on.

The input signals can reach the input of the decoder only if one of the keys of the given digital section has been pressed, closing the corresponding contact. Thus, when key 9 is pressed, the contacts  $K_{15}$  and  $K_{14}$  are closed, causing the signals  $P_1$  and  $P_4$  to pass to the output of the decoder. Similarly, when key 7 is pressed, closing the contact  $K_{12} - K_{10}$ , the signals  $P_2 - P_4$  will arrive at the output.

The signals  $P_1 - P_4$  are distributed among the decoder inputs such that the

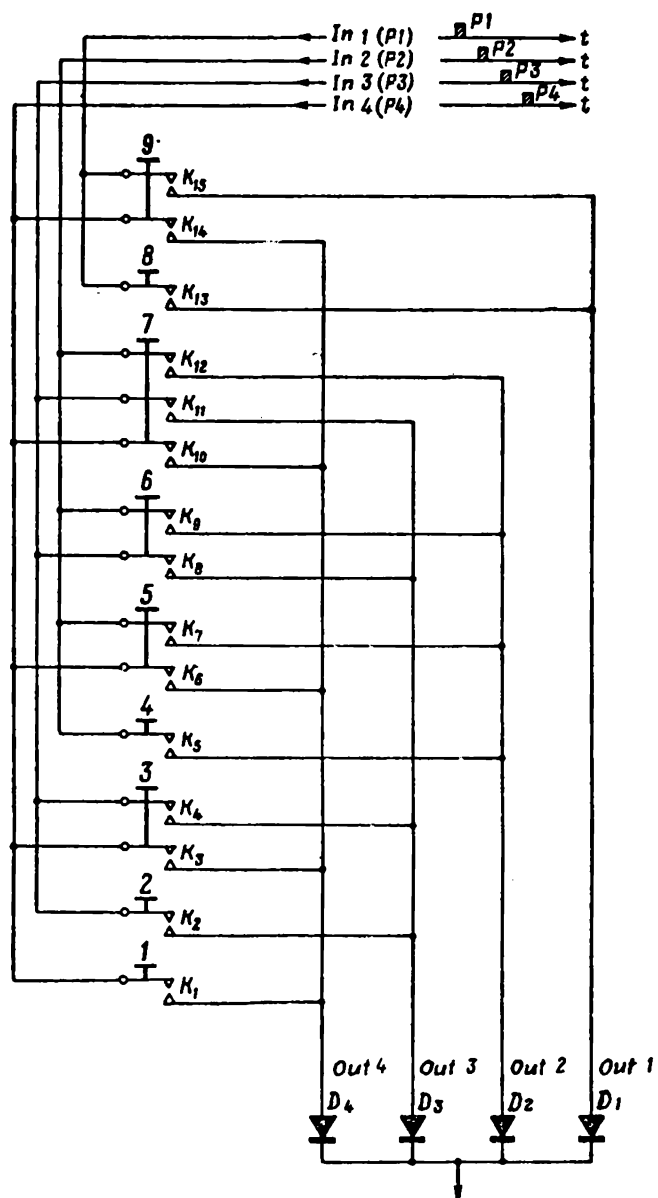


Fig.183 Schematic Diagram of Decoder of Keyboard Device in the Ural-1 Computer



signal P1 corresponds to the number 8 (1000); the signal P2 to the number 4 (100); the signal P3 to the number 2 (10); and the signal P4 to the number 1. Since these signals are also distributed in time, a sequence of signals, formed in the output circuit of the decoder when the keys are pressed, represents the serial binary code of the decimal digit to which the given key corresponds. Thus when the key 9 is pressed, the serial pulse code 1001 is obtained in the output circuit; when the key 7 is pressed, the code 0111 appears in that circuit, and so on.

The signals P1 - P4 are fed to the inputs of all the decoders of the digital sections. In their output loops are simultaneously formed the serial pulse codes representing the binary forms of the decimal digits to which the pressed keys of the digital sections correspond. The signal P2 is also fed to the sign section. It appears in the output loop of this section if the "-" key has been pressed, i.e., if the sign of a negative number has been selected.

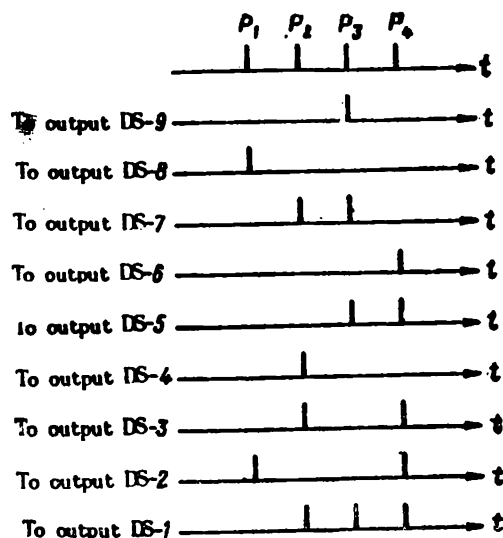


Fig.184 System of Codes for the Number 286134597

Thus, in the ten parallel output loops serial pulse codes are formed which are the binary equivalents of the sign and digits of the decimal number selected on the keyboard, or, in other words, the decimal number is represented in binary-coded decimal notation. A similar process takes place in selecting octal numbers, and the numbers are expressed in the binary system.

The pulse codes formed at the outputs of the keyboard device are transmitted to the input punch, where the corresponding system of punches is punched into the tape. These same pulse codes may be transmitted to the check-reader for checking the accuracy of the punches on the tape to be checked. [327]

Consider the operation of the building blocks of the keyboard unit in setting up and converting the decimal number 286134597. In the beginning, all the

keys are in the upper position, meaning that all decoder contacts are open, and the carriage of the printer is in the extreme right position, for printing the selected numbers.

The number is set up, with the most significant digits (MSD) first. In this case, the key 2 of the extreme left digital section is pressed first. For a negative number, the key "-" of the sign section is pressed first, followed by the keys of the digital sections. The digital and sign keys, when pressed, are fixed in the lower position, and the push-rods of the keys make contact with the corresponding contacts. In any one section, only one key at a time can be fixed in the lower position. This is accomplished by means of a simple mechanical ratchet system for blocking the keys of one section.

To the number 286134597 on the keyboard corresponds the pressing and fixing in the lower position of the keys marked on Fig.182 by the double circles. The contact  $K_2$  is closed in the decoder of the extreme left digital section, the contact  $K_{13}$  in the decoder of the next digital section, and so on. All the decoders are prepared for formation of the corresponding codes and conversion of a decimal number into a binary-coded decimal number.

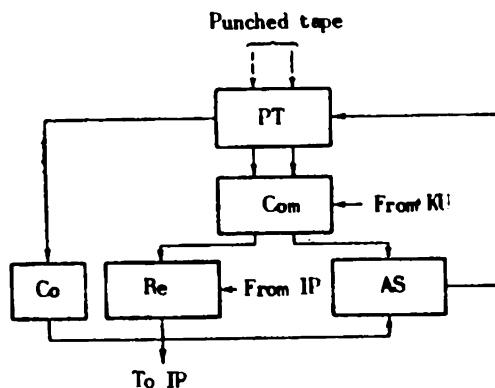


Fig.185 Block Diagram of Check-Reader

When the operate key "0" is pressed, an operate instruction is generated and transmitted to the input punch and the check-reader. On this instruction, the signals  $P_1 - P_4$  arrive from the punch at the inputs of the decoders of the keyboard device. Under the action of the corresponding pulse, codes are produced in the output circuits of these decoders. The system of codes formed when the number 286134597 is set on the keyboard is given in Fig.184. The extreme left digital section is here denoted by DS-9 and the extreme right by DS-1. The upper part of the diagram shows the time distribution of the signals  $P_1$  to  $P_4$ .

The pulse codes are transmitted from the outputs of the keyboard sections to the input punch, where they are used for punching the tape. During the entire period of formation and transmission of the codes, the operate key 0, like the depressed keys of the digital sections, remains fixed in the lower position. /328

The keys are returned to the upper position when the return key R is pressed. A blocking or inhibit system, permitting such return only after execution of the instruction to punch the tape, is connected with this key.

When the third key of the control section is pressed (Int), spaces between the digits printed by the printer on a blank are formed.

The check reader or verifier is designed to make an automatic check as to correct punching on the tape of the codes of the numbers to be introduced into the computer, and also to repunch the tape by the aid of the input punch.

This device has three main operating modes:

1. Checking mode A. In this mode, the information is simultaneously read from two identical tapes and the identity of the punches in these tapes is checked. If the holes are not the same, the motion of the tapes is automatically stopped and a non-coincidence signal is given. The amount of numbers checked is recorded by a counter.
2. Checking mode B. In this mode, a number read off one tape is compared with the number set up on the keyboard of the keyboard unit; if they agree, another tape is punched by the aid of the input punch; if they disagree, a signal is given and the punching is not performed.
3. Repunching mode. In this mode, numbers are read off one tape and another tape is punched by the aid of the input punch in accordance with the

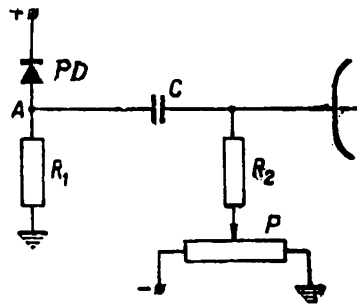


Fig.186 Connection of Photodiode to the Amplifier Input

data read off. The amount of repunched numbers is recorded by a counter.

Figure 185 is a block diagram of the check-reader. The main building blocks of this assembly are: the phototransmitter PT, the comparator Com, the automatic stop AS, the registers Re, and the counters Co.

The phototransmitter serves to transport the punched tapes to be checked, to read the number codes from them, and to produce several control signals /329

required for the joint operation of the check-reader and the keyboard mechanism and the input punch. This device consists of two identical sections. The number codes are read from the tape by 11 photodiodes (the number of code channels on the tape) so positioned that the light ray strikes each of them only if there is a hole on the corresponding channel at the site between the photocell and the light source.

The photodiodes are connected to the inputs and amplifiers, so that when the photodiodes are operating, i.e., when they are reading the codes 1 from the tape, rather powerful signals are obtained. Figure 186 shows one version of connecting the photodiode to the amplifier input. The nonilluminated photodiode has a very great back resistance while the resistance of  $R_1$  is far smaller, so that a low potential level is formed at the point A. When the photodiode is illuminated, its back resistance decreases by a factor of six to seven, causing a jump to a higher potential at the point A so long as the photodiode is illuminated. The resultant voltage pulse passes across the bypass capacitor C to the grid of the amplifier tube. The amplification is regulated by varying the bias voltage by means of the potentiometer P.

In the comparison part of the check-reader, the signals arriving from the photodiodes of the two sections in the checking mode A are compared; or the

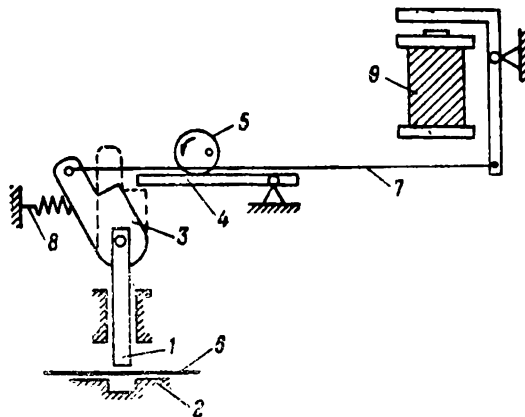


Fig.187 Punching Part of the Puncher  
 1 - Punch; 2 - Die; 3 - Pawl of punch;  
 4 - Punch strip; 5 - Eccentric;  
 6 - Punched tape; 7 - Connecting rod;  
 8 - Return spring; 9 - Punching  
 electromagnet

signals from the photodiodes of one section in the checking mode B are compared with the signals from the keyboard unit. If the compared signals disagree, a signal is produced and fed to the Com to stop all mechanisms of the check-reader.

The register part records the codes of the compared numbers, which are fed in the repunching mode to the input punch, to punch a new tape. These codes arrive from the comparison part only if they agree.

The counter part counts the amount of compared numbers. This device has two counters, each operating under the action of the signals received from the control circuit of the CR. The number of these signals equals the amount of compared numbers. When one counter is entirely filled, a signal is fed to the Com to stop the mechanisms of the check-reader. If they must be stopped after comparison of a certain amount of numbers, the corresponding complement is first fed to their counter.

The automatic stop part stops the mechanisms of the check-reader when- /330  
ever it receives a mismatch signal from the comparison part to the effect that the compared codes disagree in the checking mode A or the checking mode B, and also whenever it receives a signal from the counter part that a prescribed amount of compared numbers has already passed.

The check-reader in the checking mode A compares up to 200 numbers a minute and repunches at a speed of up to 180 numbers a minute.

The input punch is designed to automatically enter the initial numerical data and programs on the punch tape in the form of a definite system of holes. By the aid of the input punch, tape can be punched in accordance with the values of the numbers set up on the keyboard of the keyboard unit, and automatically repunched from the punched tape fed to one of the sections of the phototransmitter of the check-reader.

The punch is an electromagnetic device in which tape is punched by the aid of eleven punches as it moves between the base of the punching device (die) and a row of punches. The punching is controlled by the punching electromagnets, which operate under the action of signals from the outputs of the decoders of the keyboard assembly or from the outputs of the register portion of the check-reader.

Figure 187 shows the principle of tape punching and the connections between the main parts and the parts of the punch assembly. At first, there is no current in the winding of the punching electromagnet 9 and its armature is in the upper position. The return spring 8 holds the pawl 3 of the punch 1 in the extreme left position, and therefore when the eccentric 5 rotates only the punch strip 4 descends and, rotating about its axis, fails to engage the pawl of the punch. The tape 6 passes between the punch and the die 2; its motion is discontinuous. At the instant of punching in a given row, the tape is motionless, but after a punch it shifts one pace and stops to punch in the next /331  
row.

When the circuit of the winding of the punching electromagnet is closed, which takes place on arrival of the code signal 1 from the KU or from the CR, the armature of the electromagnet is attracted to the core and, by means of the rod 7, displaces the pawl of the punch to the extreme right position indicated by the broken line in Fig.187. Now, rotating about its axis under the action

of the eccentric, the punch strip engages the pawl of the punch and depresses it together with the punch, thus causing the punch to punch the tape. The punch and other parts are then returned to their original position under the action of return springs (not shown on the diagram) while the tape is advanced one pace.

Figure 188 gives the control circuit of one punching electromagnet. The grid of the normally cutoff tube L is fed with signals from the output of the

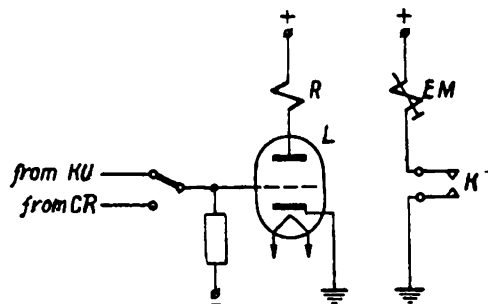


Fig.188 Control Circuit of Punching Electromagnet

corresponding decoder of the keyboard assembly or from the register portion of the check-reader. The code signal 1 opens the tube in whose plate circuit the winding of the relay R is connected. Current then flows through the relay R tripping it and thus closing the contacts K. Since these contacts are in the circuit of the winding of the punching electromagnet EM, this closing trips the electromagnet, thus also causing the tape to be punched. The code signal 0 does not open the tube L so that, in this case, the tape is not punched.

Assume that the decimal number 286134597 has been set up on the keyboard of the keyboard assembly. In this case, pulse codes are generated at the outputs of the decoders of the keyboard assembly. These codes are formed on the arrival of the control signals P1 - P4 from the punch, which signals in turn are produced on the closing of contacts connected with a special camshaft of the punch mechanism. The first one produces the signal P1; consequently, the first to appear in the punch are signals corresponding to the codes of the most significant digits (octal) of the binary-coded representations of the decimal digits set up on the keyboard.

On arrival of the control signal P1 at the KU, the signals of code 1 are fed to the inputs of tubes controlling the punching on the 8<sup>th</sup> and 1<sup>st</sup> channel of the tape. The corresponding punching electromagnets are then operated, and the tape is punched on the first row in the 8<sup>th</sup> and 1<sup>st</sup> channel, as shown in Fig.189. Thereafter, before the control signal P2 has been produced, the 1332 tape is advanced one step, and the parts of the punching device of the punch are returned to the initial position.

When the control signal P2 arrives at the keyboard assembly, the code

signals 1 are fed to the inputs of the tubes controlling the punching on the 1, 3, 4 and 7<sup>th</sup> channels; on these channels the tape is punched in the second row. After producing the control signal P3, the tape is punched similarly on channels 1, 5, 7, and 9 of the third row and, after the signal P4, on the channels 1, 2, 3, 5, and 6 of the fourth row. Figure 189 shows all these holes; the direction of tape travel is denoted by the three arrows.

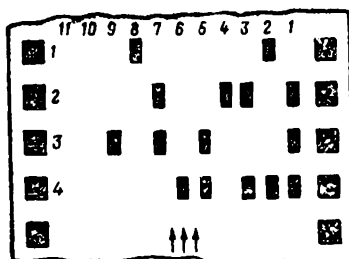


Fig.189 Punching of the Number 286134597

In this punch which operates automatically, the check-reader in the re-punching mode can punch up to 180 numbers a minute.

The tape reader is designed to read information from the tapes after preparation and checking, and to transmit this information to one of the registers of the arithmetic unit and then to the working memory of the computer. The reading is effected by a photographic method, as in the check-reader.

In the Ural-1 computer, the tape reader also belongs to the set of devices of storage on the punched tape. In fact the punched tapes can serve as a memory and storage medium for the numbers fed to the computer not only before, but also during, solution of the problem. The punched tapes used in the Ural-1 computer have a maximum length of 250 m. Up to 10,000 numbers or instructions can be punched in this length.

Data are read at fairly high speed from the punched tape when it is fed to the working memory unit of the computer: up to 4500 numbers a minute at a tape speed of 1.4 m/sec.

#### Section 49. Computer Output Equipment

The output of data from computers is mostly, as already indicated, by punched cards or tape, or by printing.

The punching of cards and tape at the output is practically the same as at the input. For this reason, the output punches differ hardly from the input devices in design and operating principle.

When the output of data from a computer is automatic, cards and tapes are

punched at relatively low speed. Thus, the output punch of the Ural-1 computer punches the codes of 150 numbers a minute and that of the Strela computer, 600 numbers a minute. These punches operate at practically the same speed, since /33 the code of one number on the tape of the Ural-1 computer is punched four rows at a time.

The results of the solution of a problem and the program taken out of a computer can be printed by various methods. The method of printing by means of

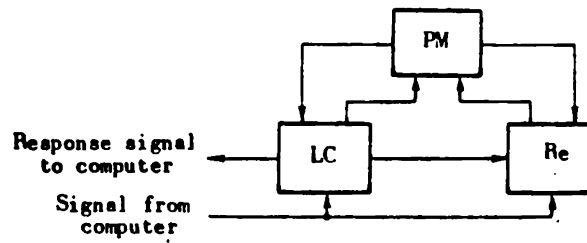


Fig.190 Block Diagram of Printer of the Ural-1 Computer

rack and drum mechanisms is most widely used in Soviet computers.

Rack-type printers. A rack-type printer automatically translates numbers from the binary-coded decimal system to the decimal system and from the binary system into the octal system, and prints them on a paper tape in the form of rows of digits. Rack-type printers of various digital computers are all based on the same general design principles. Let us consider these principles on the example of the Ural-1 printer.

This printer is a rather complex electromechanical printer with electronic control. Figure 190 shows its block diagram. The printer consists of three functional blocks: the printer mechanism proper PM, the local control unit LC, and the register building block Re.

The printer mechanism is designed to print numbers on a paper tape on receiving signals from the register and from the local control. The digits are printed by the aid of racks with 10 plungers lettered from 0 to 9. The building block has 19 racks in all, so that 19 decimal digits can be printed on a single row. A small number of these digits are generally used to indicate the exponent of the number, and the remainder to represent the number itself.

Figure 191 is a simplified schematic of part of the printer mechanism, illustrating the principle of printing by racks. The rack 1 with the plungers 4 is moved upward until the typesetting or setting electromagnet EMs is tripped. When this magnet operates, its armature and connecting rod force the stop pawl 2 to engage one of the teeth on the side surface of the rack. Thus, by means of the setting electromagnet, whose winding is fed with control signals from the registers, the stop pawl of the rack can be fixed in any desired position.

When the rack is fixed, one of its plungers will be between the hammer 3



and the printing shaft or contact drum 5, to which the paper tape 6 is pressed by a special mechanism not shown on the diagram. If now a control signal 1334 is fed to the winding of the printer electromagnet *EMp*, it will attract the armature to its core and rotate the hammer 3 about its axis. When the hammer is rotated, it will strike against the end of the printing plunger with the letter, and the corresponding decimal digit will be printed on the tape.

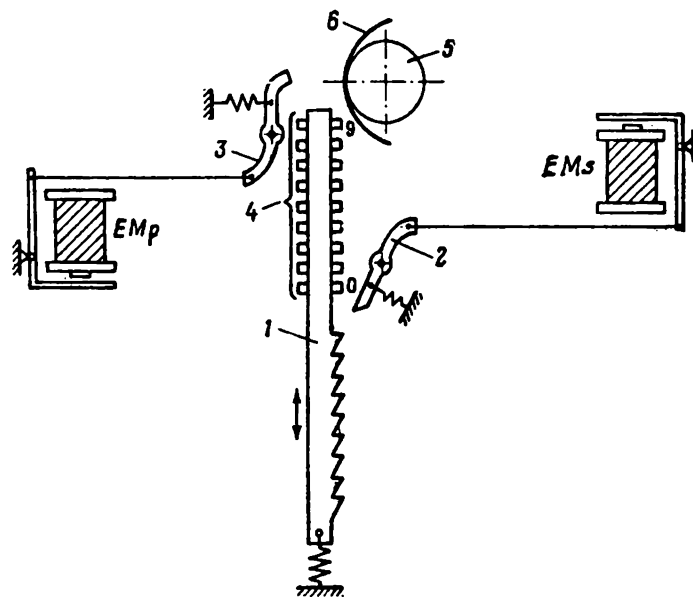


Fig.191 Rack-Type Printing Mechanism  
 1 - Rack; 2 - Pawl; 3 - Hammer; 4 - Plungers;  
 5 - Printing roller; 6 - Paper tape

After printing, the hammer and pawl are returned by the springs to their neutral positions, and the rack is lowered. All racks of the printer are raised and lowered simultaneously by a special mechanism.

The local control functional block is designed to produce signals that control the operation of the printer in accordance with the control signals from the computer, and to produce a response signal indicating the end of the printing of a given number. The signals from this building block go to the register unit for forming the ones-complement binary codes of the decimal digits, and to the printing mechanism for regulating the operation sequence of the mechanisms and of the electromagnets.

The register portion is designed to write the codes of the numbers and instructions from the computer, to count the printed numbers, and to produce signals that control the setting electromagnets. This building block comprises nine four-column binary registers to accommodate the binary-coded decimal code of the number to be printed; a binary-coded decimal counter register to count 1335 the numbers printed and to write the binary-coded decimal code of the serial number of the number being printed; an encoder; coincidence circuits; and an

inverter.

The encoder of the register block is a diode network which, on excitation of one of its input buses, forms a definite combination of output signals. The schematic diagram is shown by Fig.192.

The input buses are excited when the contacts  $K_1 - K_8$  in the local control unit are closed. When the contact  $K_1$  is closed, a high potential is established

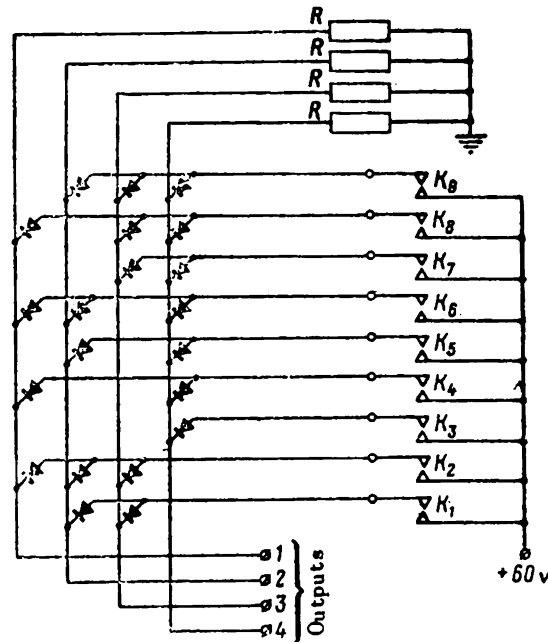


Fig.192 Encoder of Register Building Block

at the outputs 2 and 3 of the encoder, corresponding to the code 0110; when the contact  $K_2$  is closed, the code 0111 appears at the outputs; when the contact  $K_3$  is closed, the code 1000 appears, and so on. Thus, on successive closing of contacts from  $K_1$  to  $K_8$ , the inverse (ones-complement) binary codes of the decimal digits from 9 to 1 are obtained at the outputs of the encoder. The inverse code 0110 corresponds to the direct binary code 1001, i.e., to the code of the decimal digit 9; the ones-complement code 0111 corresponds to the direct binary code 1000, i.e., to the code of the decimal digit 8, and so on.

The coincidence circuits produce the control signal when certain codes coincide at their inputs. In the Ural computer, these circuits produce the control signal when the codes (direct and inverse complement) of the same decimal digit are simultaneously fed to both groups of their inputs.

The inverters are designed to invert the control signals produced by the coincidence circuits. From the outputs of the inverters, the control signals /33 are directly fed to the windings of the setting electromagnets.

Figure 193 shows a schematic diagram of the control of the setting electromagnet. The encoder Enc of the system is common to the entire register assembly. The inverse binary codes of the decimal digits formed at the outputs of the encoder go to one of the groups of inputs of the coincidence circuit CC. The binary code written in the register Re, representing the decimal digit of

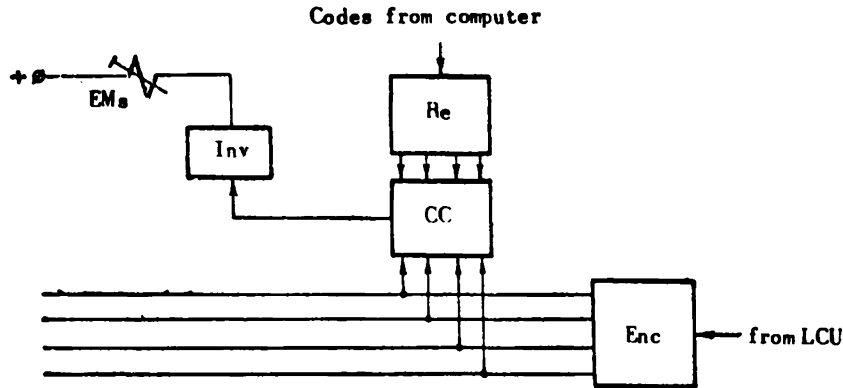


Fig.193 Control System for Setting Electromagnet

one of the places of the number put out for printing, is fed to the second group of inputs of this circuit. The control signal formed at the output of the coincidence circuit when the codes of the same decimal digit are fed to both groups of its inputs is inverted in the inverter Inv and goes to the winding of the setting electromagnet EMs.

A rack-type printer operates as follows: The number to be printed is fed from the computer in binary-decimal code to the printer and recorded in the register unit. At this time the register-counter records the binary-decimal code of the serial number of the number to be printed. Thus, the binary codes of the decimal digits of the number to be printed and its serial number are fed to the second groups of inputs of the coincidence circuits controlling the operation of the setting electromagnets.

On instruction from the control unit of the computer, the local control unit of the printer produces signals coordinating the operation of all assemblies and mechanisms during printing.

Printing is performed in several stages.

In the first stage, the number for printing is set up. On a signal from the local control, the mechanism for raising the racks begins to move all 19 racks upward. Simultaneously, the contacts  $K_1 - K_9$  connected with the input buses of the encoder are successively closed. The inverse binary codes of the decimal digits, beginning with the code of the digit 9, now arrive at the first groups of inputs of the coincidence circuits. The raising of the racks and the generation of the ones-complement binary codes of the decimal digits are so synchronized that the AND gate emits a control pulse at the instant when

the plunger with the decimal digits, corresponding to the code of the digit of the number being printed, is between the hammer and printing roller. The control pulse, arriving in the winding of the setting electromagnet, fixes the rack in the required position. Since all the racks are fixed in parallel, the number is selected or set up by the end of the first stage, i.e., the plungers with its digits are between the hammers and the printing roller.

The number is printed in the second stage. A signal is sent from the local control unit into the windings of all the printing electromagnets, and under its action the electromagnets are tripped, causing them to print all the digits of the number and its serial number on a single row of the tape.

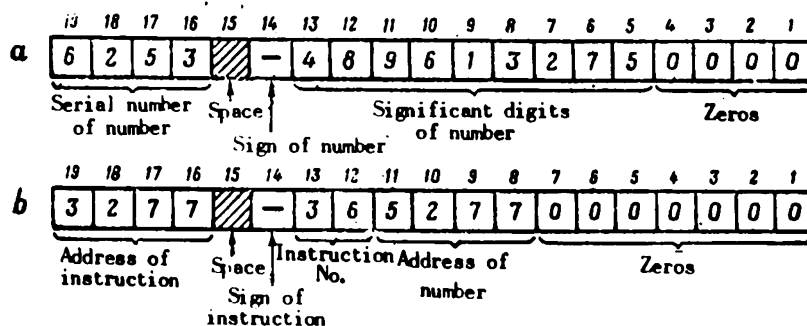


Fig.194 Printing  
a - Of numbers; b - Of instructions

In the third stage, all parts and mechanisms of the printer are reset by return springs to their neutral positions, and a signal that printing is completed is sent to the local control unit. On this signal, a response printing signal is fed to the control unit of the computer, indicating that the printer is now ready to print the next number. On completion of the printing, a signal is fed from the printing mechanism to the register unit in which the binary-decimal code of the serial number of the number to be printed is increased by 1.

The instructions from the computer are similarly printed. The only difference is that they are not printed in decimal but in octal notation, arranged in a different way along the row of the tape. Figure 194 shows the arrangement of numbers and instructions in printing the rows of the tape. The zeros need not be printed, and to avoid doing so, the corresponding racks are stopped.

A rack-type printer operates at low speed, about 100 numbers a minute. /338

Drum-type printer. A drum-type printer, like a rack-type printer, is an electromechanical printing system with electronic control but differs in the method of printing the numbers and instructions received from the computer.

Such printers are used in the Ural-2, BESM-2, and other computers. They print up to 1200 numbers a minute (20 numbers a second) which considerably

shortens the time for output of the results of the solution from the computer.

Figure 195 is a general schematic of a drum-type printer. It comprises the digit drum DD, the code drum CD, the functional block of printing electro-magnets EMP, the photocell building block PC, the four-column binary registers Re, the coincidence circuits CC, and the amplifiers A.

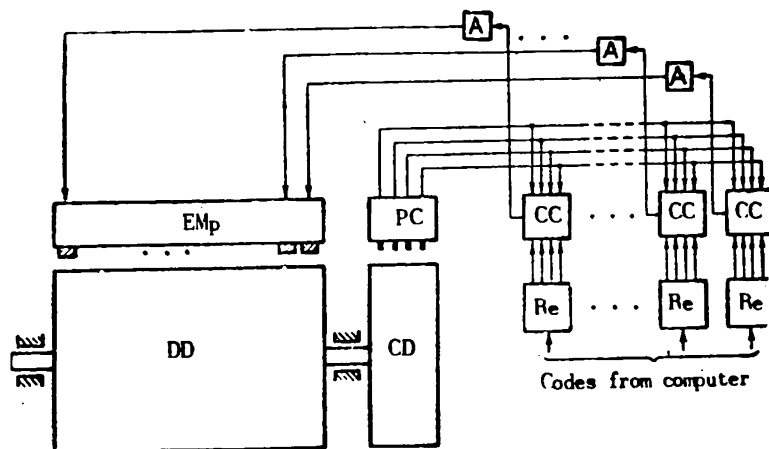


Fig.195 General Schematic of a Drum-Type Printer

The digit drum comprises individual digit wheels, each engraved with the digits from 0 to 9 and the signs "+" and "-". The digit wheels are mounted to a shaft in such a way that the single characters are arranged in lines (generatrices of the drum). The lines with the characters occupy about 5/6 of the outer surface of the drum; the remainder is blank. During operation of the printer, the digit drum and the code drum, seated on the same shaft, rotate continuously at a speed of 20 rps.

The code drum serves to form the binary codes of the decimal digits and signs whose characters are placed on the digit drum, together with the series of sync and control pulses. The codes of the digits and signs are arranged 1339 in columns forming a system of openings in the outer circumference of the code drum, with the code 1010 corresponding to the "+" sign and the code 1011 to the "-" sign. Each line, along the fifth channel, has openings for the sync pulses SP. There are also openings for the signals "start printing" and "stop printing". A light bulb is mounted in a fixed position inside the code drum.

The columns of the two drums are coordinated, meaning that one column of the digit drum carrying some decimal digit corresponds to one column of the code drum carrying the binary code of the same digit. Figure 196a shows the development of the surface of the digit drum and Fig.196b that of the code drum. The holes corresponding to the least significant digit of the binary codes of the digits and signs are made to coincide with the first channel on the code drum, on the left-hand side.

The functional block of printing electromagnets acts as the final control element of the printer. It directly prints numbers and instructions from the computer, and is provided with a series of electromagnets whose armatures are 134

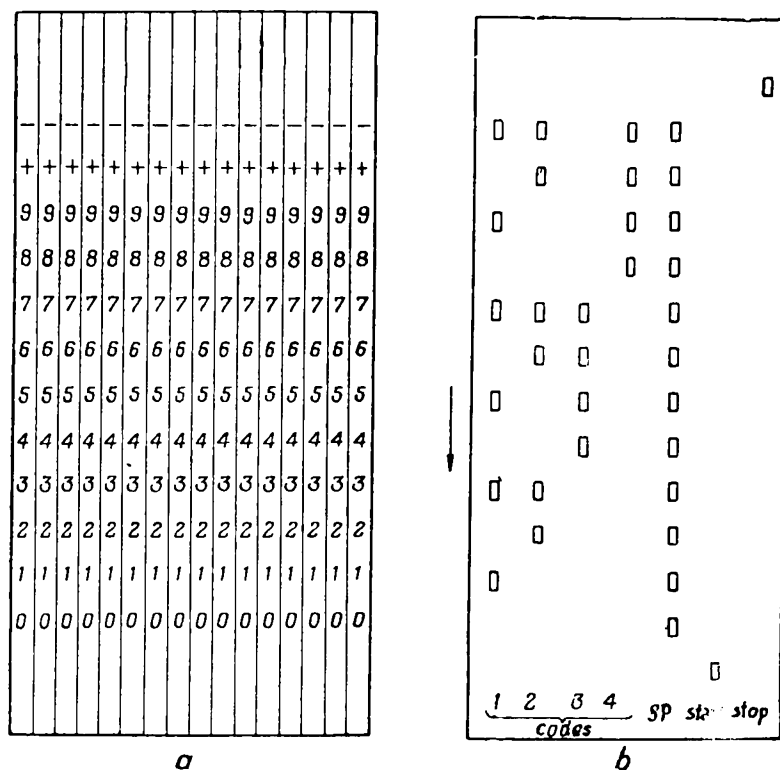


Fig.196 Development of Drum Surfaces  
a - Digit drum; b - Code drum

attached to printing hammers. There are as many electromagnets, and therefore as many hammers, as there are channels on the digit drum. In their initial state, all hammers are off the tape, which latter is positioned between the hammers and the digit drum as shown in Fig.197. When a control signal is fed to the winding of the printing electromagnet EMP, the armature is attracted, the hammer 2 strikes the tape 3 forcing it against the digit drum 1 which, at that instant, is positioned under the hammer.

The photocell assembly is used to read the codes off the code drum and transfer them to one of the groups of inputs of the coincidence circuits. Together with the code drum, it performs functions similar to those of the encoder with cam contacts in the rack-type printer.

The four-column binary registers are used to record the binary-decimal codes of the numbers to be printed and their ordinal numbers, together with the codes for the sign digits. The codes of instructions are also entered in these registers.

The coincidence circuits, as in the rack-type printer, are designed to produce control printing signals on coincidence, at the two groups of inputs, of the codes from the photocell assembly and from the corresponding registers.

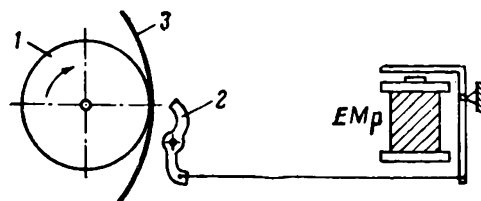


Fig.197 Schematic of Printing from Digit Drum  
1 - Digit drum; 2 - Hammer; 3 - Paper tape

These signals are then amplified by a group of amplifiers and routed to the windings of the printing electromagnets.

The printer operates as follows: Before beginning to print, the code of the number is fed to the registers and recorded there. All hammers are withdrawn from the tape, which is not advanced when a single number is printed. When the digit drum and the code drum rotate, the binary codes of the decimal digits are successively produced in the photocell unit and fed to the coincidence circuits.

The operations of the individual functional blocks of the printer are so synchronized that the photocell block feeds the AND gates with the binary code of the digit whose column is beneath the hammers at the given instant. If the same digit is present also in any of the columns of the number to be printed, then the corresponding coincidence circuit produces a control signal which, on entering the winding of the printing electromagnet, will cause the printing of that digit.

All rows of digits pass under the hammers, and all rows with code holes pass under the photocells, in  $5/6$  of a revolution of the drum. During this 1/341 period, the binary codes of all decimal digits appear at the outputs of the AND gates, and therefore all digits of the number will be printed at the corresponding instants of time.

Thus, a number is printed in  $5/6$  of a revolution of the drums. During the remaining sixth of the revolution, the unit prepares to print the next number. The parts and assemblies of the printing electromagnet functional block are returned to their neutral position, the code of the next number and its ordinal number are fed into the registers of the printer, and the tape is advanced one space, i.e., to the next row.

Section 50. General Propositions on the Conversion of Continuous Magnitudes into Discrete Quantities and of Discrete Quantities into Continuous Magnitudes

The conversion of continuous magnitudes into discrete quantities and of discrete quantities into continuous magnitudes is required whenever a digital computer is used in a control system. In order to produce the control actions,

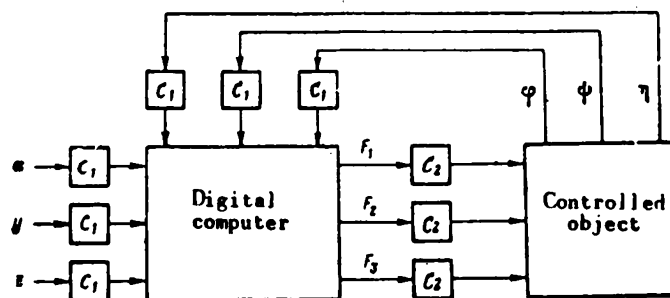


Fig.198 Control System with Digital Computer

the digital computer must receive information on the position of the controlled object or the course of the controlled process, and on various external conditions. This information is usually represented in the form of continuous quantities which, before input to the computer, must be converted into discrete quantities, namely, into the corresponding number codes.

To calculate the control actions, the digital computer performs operations according to the information received in the form of discrete quantities; the result of the calculation is likewise represented in the form of discrete quantities. For the control actions to be sensed by the final control elements, the discrete quantities must in most cases be converted into continuous quantities at the computer output.

Figure 198 gives a block diagram of a control system with a digital computer. The quantities  $x$ ,  $y$ , and  $z$ , arriving at the computer input through 342 the group of converters  $C_1$ , characterize the external conditions affecting the control process of the object. These quantities are usually continuous, which means that the converter  $C_1$  must convert them into discrete quantities.

Information on the position or state of the controlled object arrives in the form of the continuous quantities  $\varphi$ ,  $\psi$ , and  $\eta$ . Before input to the computer, these quantities are converted into discrete quantities by a second group of converters  $C_1$ .

If the values of  $\varphi$ ,  $\psi$ , and  $\eta$  differ at a given time from the required or optimum values, then the computer calculates and feeds to the output the discrete quantities  $f_1$ ,  $f_2$  and  $f_3$ , which the converters  $C_2$  convert into continuous quantities and feed to the controlled object. The converted quantities  $f_1$ ,  $f_2$ ,



and  $F_3$  are control actions, which compel the final control elements of the object to assume the required position or set it into the required state.

The continuous quantities by means of which information is fed to the control systems from a digital computer, are usually voltages of an electric current, or else displacements; the displacements in many cases are represented in the form of the rotation of read shafts through some angle. The discrete quantities with which the computer operates are numbers, for which reason typical converters of continuous quantities into discrete quantities include devices of the "voltage-to-number" and "shaft-to-number" types (voltage digitizer and shaft-digitizer).

The process of conversion of a continuous quantity into a number is divided into two stages: quantization, or discretization of the continuous quantity, and encoding.

It is often difficult to differentiate between these stages in the general conversion process, but their distinctive features can always be indicated.

By quantization we mean the representation of a continuous quantity in the form of a finite number of discrete elements or states: potential (voltage) levels, pulse sequences, numbers of excited buses of some device. The procurement, by quantization, of a finite number of potential levels in the form of a certain step function, a finite sequence of pulses, or a finite number of excited buses, facilitates the subsequent encoding.

The methods of encoding depend on the systems in which the codes are to be used. For digital computers, digital coding is the most acceptable; in such codes, a certain number corresponds to each combination of discrete elements or states. Since digital computers generally use the binary system of notation, such encoding gives the codes of binary numbers.

It is simplest to encode a pulse train: The pulses are counted by a 1343 binary counter, and the state of its flip-flops after the count will express the corresponding binary number. A finite number of potential levels or "step-lets" can be encoded by differentiation of this complex step signal, forming a pulse train numerically equal to the number of steplets in the signal. These pulses are then counted by a binary counter.

If, as the result of quantization, a finite number of buses are excited, encoding is done by the aid of encoders: The buses to be excited are connected to the encoder inputs, and the corresponding codes of the binary numbers are formed at their outputs.

A large number of converters of various types have been developed and are in use. Details and the rather complex classification are discussed in the specialized literature (Bibl.6, 9). To simplify the exposition of the fundamental principles of the converter circuitry, such converters can be subdivided into three main groups: time systems, indirect encoding systems, and direct encoding systems.

The time systems include converters with intermediate conversion of a continuous or a discrete quantity into the proportional time interval. The time interval obtained as a result of this intermediate conversion is then measured by means of a quartz oscillator and a binary counter (in conversion of continuous quantities) or is generated by special servosystems when discrete quantities are to be converted.

Indirect encoding systems likewise effect an intermediate conversion of the continuous quantities, but without generating the corresponding time intervals. Thus the angle of shaft rotation may first be converted into a pulse train, after which the pulses are counted by a binary counter.

Direct encoding systems, as a rule, include converters of continuous quantities into discrete quantities. They do not perform intermediate conversions but directly convert voltages or displacements (angle of shaft rotation) into the corresponding digital codes.

Converters of discrete quantities into electrical voltages are designed primarily on the "weighting" principle. Each digit of a binary number is correlated with a definite value of a continuous quantity (current or voltage). The continuous quantities, "weighted" by a binary law, and correlated with the digits of the number to be converted, are summated, giving a voltage at the converter output proportional to the initial number.

Converters of discrete quantities into angular displacements (angles of shaft rotation) may be feedback or non-feedback systems, with the feedback systems giving higher accuracy of conversion. The feedback converters ordinarily used are digital servosystems, with other types of converters of discrete and continuous quantities as components.

## Section 51. "Voltage-To-Number" Converters

Converters of DC voltages, or devices of the "voltage digitizer" type, are designed most simply by using the principles of time systems or time-encoding

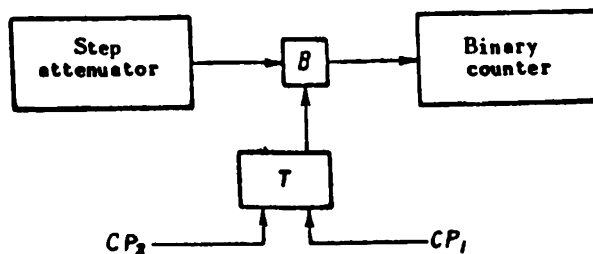


Fig.199 Block Diagram of Time-Encoding Conversion System

technique. In this case, the voltage is converted into the proportional time interval, which is then measured. For the case that this interval is limited

by certain pulses, the conversion principle can be illustrated by the aid of Fig.199.

The pulses from the step attenuator arrive at the counter input only when

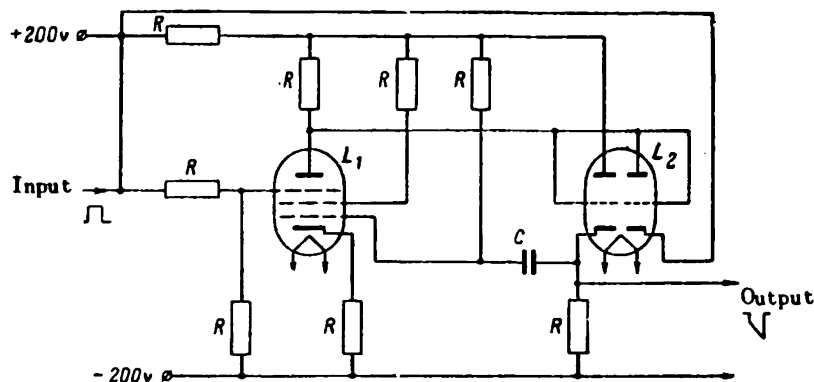


Fig.200 Phantastron

the flip-flop T is in the state 1 and opens the gate B. The flip-flop is flipped to this state by the control pulse  $CP_1$ , which initiates the time interval. The flip-flop is reset to the state 0 by the control pulse  $CP_2$ , which

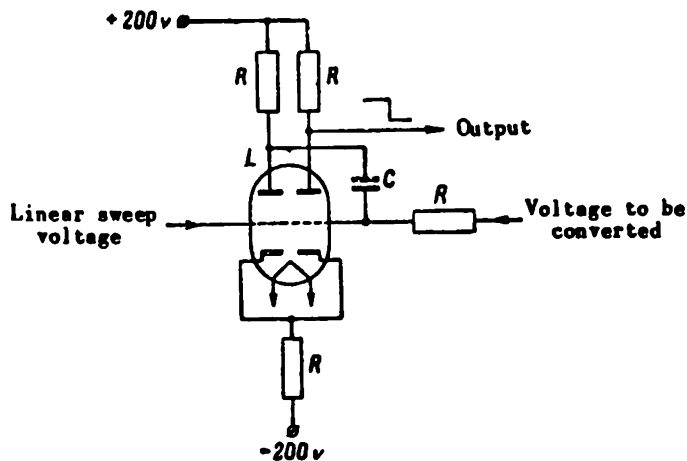


Fig.201 Comparator Circuit

is fed at the end of the time interval; this inhibits the gate and stops the flow of pulses to the counter input. The number of pulses counted by the counter will be the discrete quantity into which the continuous quantity (here, the voltage) is converted.

This diagram is incomplete; it does not show the devices for generating

the control pulses that clock the beginning and end of the time interval. The conversion of voltage into a time interval, and the generation of the initial and final control pulses, i.e., the principal operations involved in the quantization of the continuous quantity, are performed by the linear sweep oscillator and the comparison circuit.

The linear sweep-voltage generator generates a sawtooth voltage which is fed to one input of the comparison circuit. The voltage to be converted is applied to the other input. When the voltages applied to the two inputs of the voltage comparator are the same, a pulse is formed at the output and is then used in the converter as the pulse to clock the end of the time interval. /345

The phantastron, whose circuit is given on Fig.200, is used as the linear sweep generator. A trigger pulse is fed to the input of the phantastron to excite it. The time of arrival of this pulse at the phantastron input determines the beginning of the sweep.

In the comparator circuit, a double triode may be used. Figure 201 shows one of the variants of the circuit designed to compare the sawtooth voltage from the phantastron and the voltage to be converted. As soon as these voltages

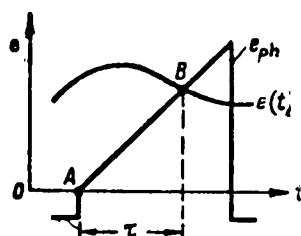


Fig.202 Generation of Time Interval

coincide, the potential at the output of the circuit will drop, thus producing the corresponding pulse. /346

Figure 202 graphically illustrates the process of voltage conversion into a proportional time interval. Here,  $e(t)$  denotes the voltage to be converted and  $e_{ph}$  the voltage from the phantastron output. The beginning of the time interval  $\tau$  during which the voltage  $e(t)$  will be converted, is determined by the beginning of the sweep A, while the end of the interval is given by the time of coincidence of the voltages which corresponds to the intersection of the curves at point B. In the practical general conversion scheme, the beginning of the time interval will be determined by the pulse triggering the phantastron, and the end by the pulse from the output of the comparator.

The general scheme of a voltage digitizer, based on the time-encoding principle, is shown in Fig.203. By means of the phantastron and the comparison circuit CC, the continuous quantity - the voltage - is converted into a time interval, whose beginning corresponds to the trigger pulse from the output of the impulser Im and whose ending corresponds to the pulse formed at the output

of the comparator. The trigger pulse is also fed to the flip-flop T, flipping it into the state 1. The flip-flop is reset to the state 0 by the pulse produced at the output of the comparator. While the flip-flop is in state 1, the gate B is open and permits passage, from the output of the step attenuator SA, of a pulse train to the counter. As a result of the pulse counting, a number is set up in the counter which is the discrete equivalent of the voltage being converted.

The accuracy of analog-to-digital conversion of voltage in this circuit is primarily determined by the linearity of the sweep. The conversion errors are

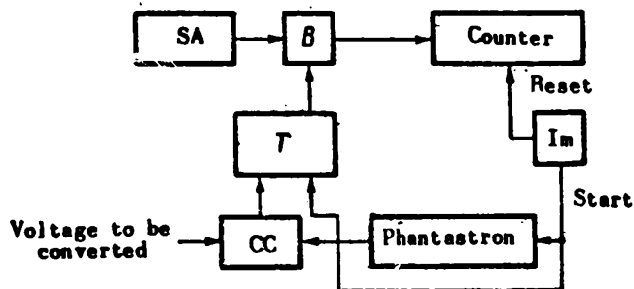


Fig.203 Schematic of "Voltage Digitizer" with Phantastron

greater, the more the sawtooth voltage from the phantastron output deviates from an ideal linear sweep. Moreover, the repetition rate of the pulses fed to the counter from the SA, and the method of reading a number from the counter, also affect the accuracy of conversion.

If the repetition rate of the pulses from the SA output is low, then, during the time the gate B is open, a small number of pulses arrive at the counter input, and the converted voltage will be represented by too small a number. With increasing pulse repetition rate, this number will increase. Since the voltage itself usually represents a certain number of physical reference units, the number into which it is converted must be such as not to decrease the accuracy of representation. In other words, if the voltage represents M reference units, then the number into which it is converted must be 1347 at least not smaller than M.

On this basis, the required SA frequency is

$$f_{sen} = \frac{M_{max}}{T_{ph}}$$

where  $f_{sen}$  is the SA frequency;

$M_{max}$  is the maximum number of physical reference units that can be represented by the voltage being converted;

$T_{ph}$  is the duration of the linear sweep of the phantastron.

Let  $T = 10$  msec, and the voltage to be converted, in the extreme case, be represented by 5000 physical reference units. Then the step attenuator should have the frequency:

$$f_{gen} = 500\,000 \text{ cycles} = 0.5 \text{ Mc},$$

The method of reading the number from the counter affects the accuracy of conversion as follows: The number formed in the counter represents the converted voltage at the instant of coincidence of the voltages in the comparator. Consequently, for accurate operation of the converter, the number must be read from the counter at the time the pulse is formed at the output of the comparator. At this instant, however, the transients may not yet have decayed so that there will be an error in the number read from the counter. We find that such an exact selection of the time of reading a number will produce an error in the number itself, since it may well be that its value has not yet been set up in the counter.

If, however, the number is taken from the counter after the end of the linear sweep cycle, then, if the voltage to be converted fluctuates rapidly, there will be an error due to the noncoincidence of the reading time with the time of voltage coincidence. The resultant number will represent the converted voltage with a certain lag, which must be taken into account in practical circuitry.

## Section 52. "Shaft-to-Number" Converters

1348

The representation of physical quantities in the form of the corresponding angles of rotation of mechanical shafts is quite common in modern technology. Devices converting from shaft rotation to numbers or so-called "shaft position

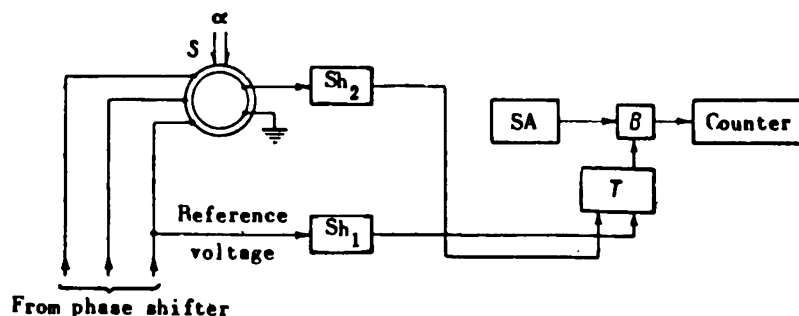


Fig.204 Single-Channel "Shaft-Position Digitizer" with Phase Shifter

digitizers" are therefore an important type of converters of continuous into discrete quantities.

Time-encoding systems. The most characteristic time-encoding systems are converters of the shaft-digitizer type with a phase shifter. Here, the shaft

position is converted into the proportional time interval by means of phase shifters, which shift the phase of the output voltage relative to the reference voltage, proportionally to the angles of rotation of the shafts. The phase shifter used may be a selsyn with a three-phase stator winding. The shaft position can be converted into a digital value by phase-shift converters on one channel or on two: the coarse reading channel and the fine reading channel.

Figure 204 shows the circuit of a single-channel converter with the selsyn, in a phase-shifter circuit, comprising the selsyn S, the shapers  $Sh_1$  and  $Sh_2$ , the step attenuator SA, the flip-flop T, the gate B, and a counter. Pulses are formed in the shapers as soon as the sinusoidal voltages applied to their inputs change from minus to plus. A pulse from the output of the shaper  $Sh_1$

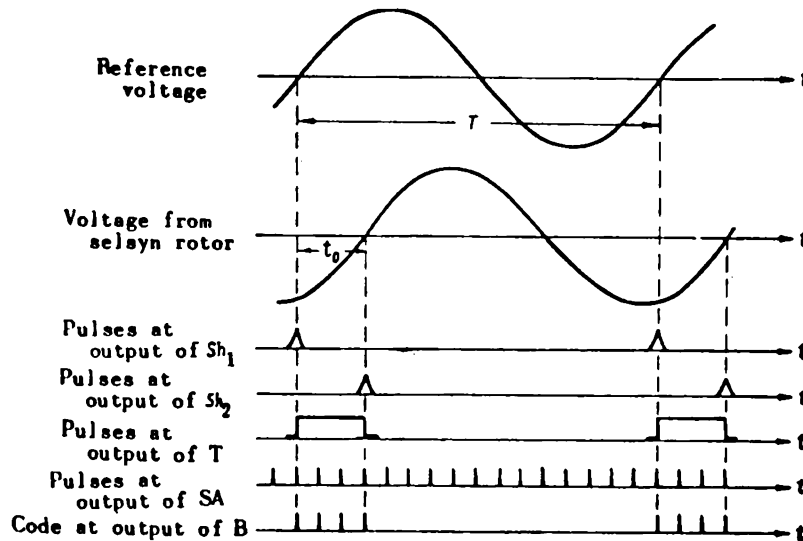


Fig.205 Timing Chart of One-Channel Converter with Phase Shifter

opens gate B via the flip-flop T, while the pulse from the output of the shaper  $Sh_2$  cuts off that gate.

When the selsyn rotor is rotated through a certain angle  $\alpha$ , its output voltage is shifted in time (in phase) relative to the reference voltage by some quantity  $t_0$ , proportional to  $\alpha$ :

$$t_0 = \frac{T}{2\pi} \alpha,$$

where T is the period of the reference voltage.

Obviously,  $t_0$  is the time interval representing the intermediate quantity in the transition from the shaft position to the corresponding digital number. 1349

Conversion of the shaft position to a digital number is illustrated in Fig.205. The rotation of the selsyn rotor through the angle  $\alpha$  causes a shift of its output voltage relative to the reference voltage by an amount  $t_0$ , which is measured by means of the system SA-gate-counter. The SA generates clock pulses that are counted by the counter during the time  $t_0$ , for which the gate is open. Thus, by the end of the time interval  $t_0$ , the number that is the discrete equivalent of the angle of shaft rotation  $\alpha$  has been set up in the counter.

To obtain the required accuracy of shaft-to-digital-number conversion, the frequency of the reference voltage  $f_r$  and the frequency of the clocks  $f_c$  (the pulses from the SA) must be correlated in a certain way. Let us assume that one rotation of the input shaft represents  $N$  physical reference units, and that

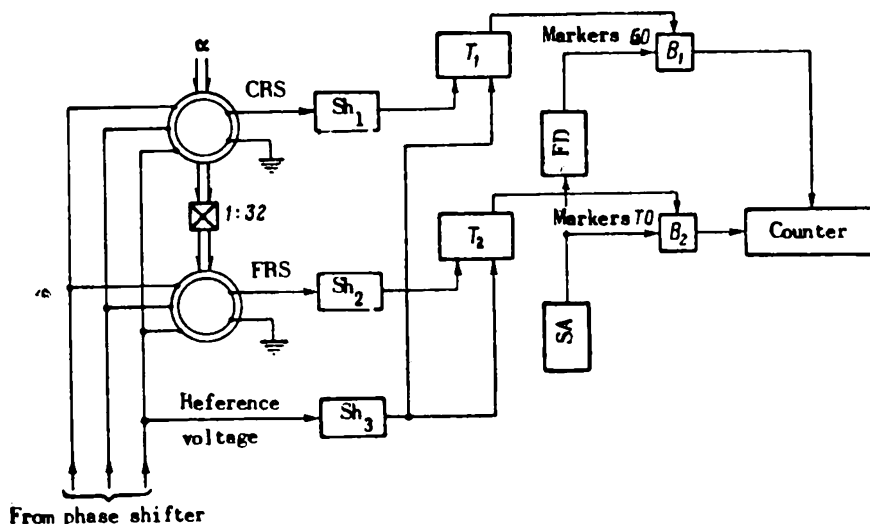


Fig.206 Two-Channel "Shaft-Position Digitizer" with Phase Shifters

to each such unit there corresponds a unit of the number into which the shaft position is converted. This condition is satisfied if, during the time  $T$  corresponding to one rotation of the input shaft, the generator SA generates  $N$  pulses. Therefore,

$$N = f_c T.$$

Since  $T = \frac{1}{f_r},$

$$f_c = f_r N.$$

Let  $N = 20,000$  and  $f_r = 100$  cycles, which is entirely realistic. Then the frequency  $f_c$  must be 2 Mc. /35

It is rather difficult to make a converter operate stably at such a high frequency. Frequency matching does not guarantee accurate operation of a one-



channel converter, in which the error of individual elements, especially those of the selsyn phase shifters, has a marked effect. For large values of  $N$ , therefore, two-channel converters are used.

A two-channel converter uses two selsyn phase shifters: the coarse reading selsyn CRS and the fine reading selsyn FRS, which are connected at a certain transmission ratio. The angle of rotation of the input shaft is converted into a digital number by the same principle as in the one-channel converter, but separately on the channels of coarse and fine reading.

Figure 206 gives a general schematic of one variant of the two-channel converter. The selsyns of coarse and fine reading are connected at a transmission ratio of  $1 : 32$ , i.e., for one rotation of the CRS rotor, the FRS rotor makes 32 revolutions. This transmission ratio was selected because it represents a power of two, thus simplifying the circuit of the converter and the matching of the fine and coarse reading channels.

The shaft angle of each of the selsyns is converted into a digital number in a manner similar to that used in the one-channel converter. The shapers  $Sh_2$ ,  $Sh_3$ , the flip-flop  $T_2$ , and the gate  $B_2$  participate in the conversion on the fine-reading channel, and the shapers  $Sh_1$ ,  $Sh_3$ , the flip-flop  $T_1$ , and the gate  $B_1$  cooperate on the coarse-reading channel. The marker pulses  $TO$  are fed <sup>/351</sup> to the gate  $B_2$  directly from the step attenuator  $SA$ , while the gate  $B_1$  is fed with the marker pulses  $GO$  from the frequency divider  $FD$ , which reduces the frequency of the pulses from the  $SA$  to the desired level. The number into which the angle  $\alpha$  is converted is in general set up for the entire converter by the counter which counts the pulses arriving from the gates  $B_1$  and  $B_2$ .

Figure 207 gives a timing chart for the two-channel converter. The chart shows that to one period of the reference voltage  $T$  there correspond 32 marks  $GO$  and 256 marks  $TO$ . The number of  $TO$  marks may differ; but in the case shown on the chart, one revolution of the input shaft represents 8192 physical reference units, and to each of them corresponds a unit of the binary number formed in the counter.

In the case under discussion, 256 markers correspond to one revolution of the FRS rotor. This means that, in any position of the rotor, not over 256 pulses can pass to the gate  $B_2$ , which is insufficient to represent 8192 physical reference units. In addition, the number of pulses passing the gate  $B_2$  does not unambiguously define the input shaft position (its angle of rotation  $\alpha$ ) since the count on the  $TO$  channel begins again after each revolution of the FRS rotor, corresponding to only  $1/32$  revolution of the input shaft. This makes it necessary to introduce additional elements to count the number of revolutions of the FRS rotor, i.e., for a unique determination of the input shaft position.

The number of revolutions of the rotor of the FRS is counted in the coarse reading channel. For each revolution of the input shaft the CRS rotor will make  $1/32$  revolution, and the FRS rotor one full revolution, corresponding to the appearance of one pulse at the output of gate  $B_1$ . Thus, the number of pulses at the output of the gate  $B_1$  will indicate how many revolutions are made

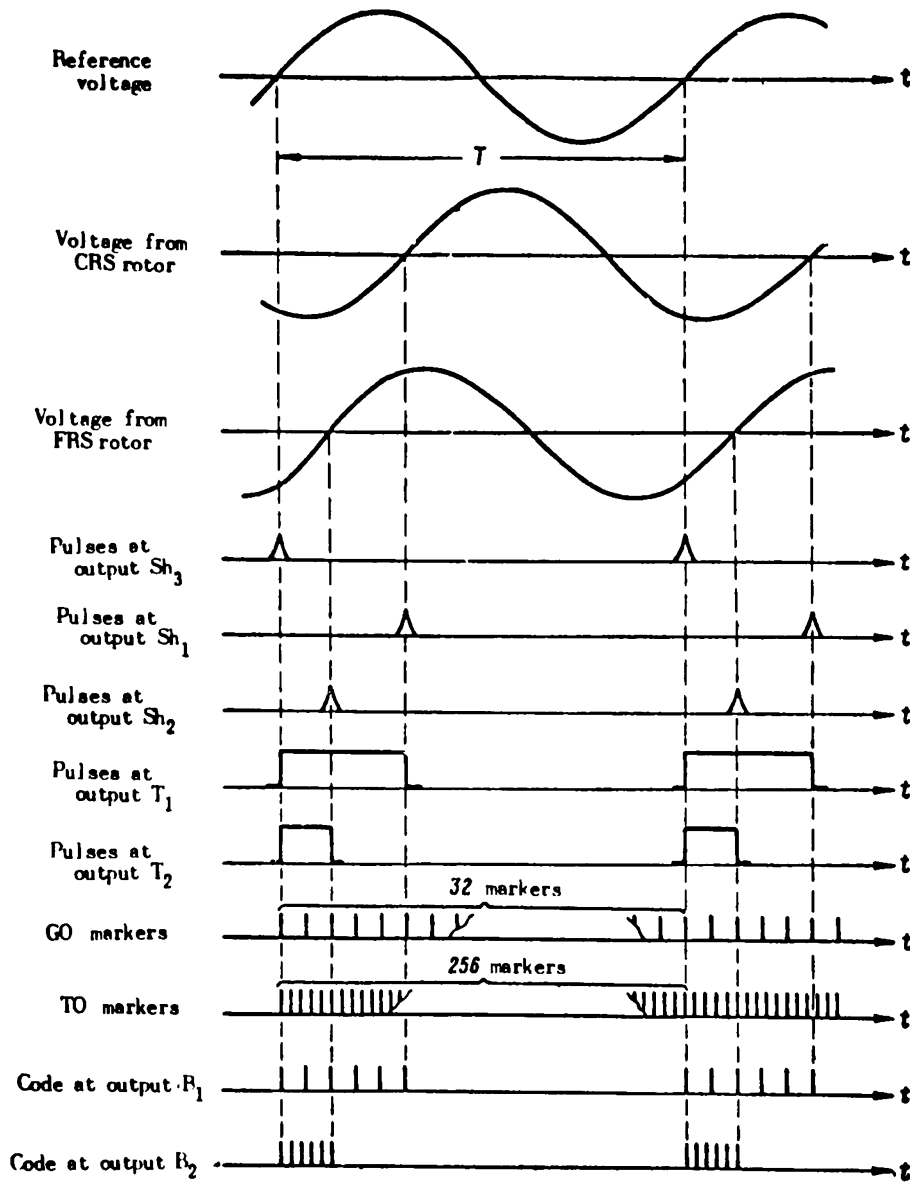


Fig.207 Timing Chart for Two-Channel Converter with Phase Shifters

by the fine reading selsyn rotor during variation of the input shaft angle from 0 to  $360^\circ$ . One GO mark corresponds to 256 TO marks, permitting ready matching of the operation of the counter elements which count the pulses simultaneously arriving from the inputs of gates  $B_1$  and  $B_2$ .

The converter uses a 13-column flip-flop counter, which makes it possible to express each unit of the number formed in it by 8192 physical reference units. The pulses from the output of gate  $B_2$  arrive at the input of the flip-flop of the least significant column of the counter. The pulses from the output of gate  $B_1$  are fed directly to the input of the flip-flop of the ninth column of the counter, corresponding to the weighting ratio of the markers of coarse and fine reading.

Let the output shaft rotate through an angle  $\alpha = \frac{5.5}{32} 360^\circ$ , and let the angle  $\alpha$  at this instant be converted into a digital number. Obviously, the CRS rotor will rotate  $\frac{5.5}{32}$  revolution from the zero position, and the FRS /353 rotor 5.5 revolutions, which practically corresponds to its rotation by 0.5 revolution. The gate  $B_1$  will be open for the time  $t_1 = \frac{5.5}{32} T$  during which 5 GO marker pulses will be passed by it to the counter. The gate  $B_2$  will be open for the time  $t_2 = 0.5 T$  during which it will pass 128 TO marker pulses to the counter. The binary number 0010110000000, corresponding to the decimal number 1408, is ultimately set up in the counter. This conversion satisfies the requirement, since the angle of rotation  $\alpha = \frac{5.5}{32} 360^\circ$  represents 1408 physical reference units under the prescribed condition that a complete revolution of the input shaft corresponds to 8192 physical reference units.

A very high frequency is selected for the reference voltage, which means that the selsyn rotors can be considered stationary during the conversion. However, the frequency must not be too high, since the proportional time intervals would become too short, and the shaft positions being converted would be represented by small numbers (at equal frequency of the GO and TO markers).

In a two-channel converter with phase shifters, as in the one-channel converter, the frequencies of the clocks (the pulses from the SA and the FD) must be matched with the frequency of the reference voltage. The frequency of the TO markers, taken directly from the output of the SA, is

$$f_1 = \frac{f_r N}{Q},$$

where  $f_r$  is the frequency of the reference voltage;

$N$  is the number of physical reference units represented by one revolution of the input shaft;

$Q$  is the transmission ratio from FRS to CRS.

The frequency  $f_2$  of the GO markers, taken from the output of the frequency divider FD, depend on the values of  $f_1$ ,  $N$ , and  $Q$ :

$$f_2 = \frac{f_1 Q^2}{N}, \text{ or } f_2 = \frac{f_1}{\frac{N}{Q^2}}.$$

If  $N = 8192$ ,  $f_1 = 100$  cps and  $Q = 32$ , then  $f_1 = 25,600$  cps and  $f_2 = \frac{f_1}{8} = 3200$  cps.

Shaft-position digitizers with ordinary phase shifters, manufactured along the line of electric machines, cannot give very high conversion accuracy even in the two-channel version, because the mechanical reducers used are a source of errors that cannot be compensated in full-scale circuits. In the extreme 35 case, converters with ordinary phase shifters can convert the shaft positions into binary numbers with not more than 13 true places. Another disadvantage of such devices is their rather great bulk and considerable weight.

Multipole phase shifters of special design are used to build relatively small and highly accurate converters for shaft positions to binary codes. In such phase shifters, the phase of the output voltage varies at a far greater rate than the angular displacement of the rotor shaft. These are known as five-speed angular displacement transmitters and may be built either capacitive or inductive. Inductive high-speed transmitters are usually called inductosyns.

An inductosyn is a multipole inductive angular-displacement pickup from which the angle can be read to within an accuracy of several seconds of arc. In contrast to the phase shifters of the electric machine type, they contain no iron and their windings are of special design, usually of the printed type. A two-channel converter can be run on a single inductosyn, because of the internal electric reduction. If properly designed, inductosyn converters can convert shaft positions into binary numbers with as many as 18 true places.

Direct encoding systems. In direct encoding systems the shaft rotation angle is converted directly into the parallel (or serial) code of a binary number. The principal elements of such systems are usually encoding disks and encoding drums, connected with electromechanical, photoelectric, or inductive systems of data reading.

Encoding disks and encoding drums are based on the principle of subdividing their surfaces into a series of fields, some of which represent the code 0 and some the code 1. The system of subdivision is such that to each position of the input shaft connected to the disk or drum there corresponds a definite combination of fields representing the code 0 or 1. Thus, at a given angle of rotation of the input shaft, a fully determinate binary number can be read off.

Consider the example of a shaft-digitizer with a four-column encoding drum and an electromechanical system of data mapping. Figure 208 is a schematic of such a device. The encoding drum CDr is divided into four parts, corresponding to the four places of the binary numbers. Its surface consists of a system of conducting and nonconducting areas corresponding to the codes of the binary

numbers. The right side of Fig.208 shows the developed surface of an encoding drum; the dark areas are nonconducting or insulating and the unshaded areas are conducting.

All the conducting areas are electrically connected with the command bus CB to which the control pulses are fed from the control unit CU. A pulse arriving at the CB can proceed to a given input of the amplifier-shaper unit ASU only if at least one of the electric brushes  $Br_1 - Br_4$  makes contact with a conducting area on the drum surface. If several brushes are at the same time

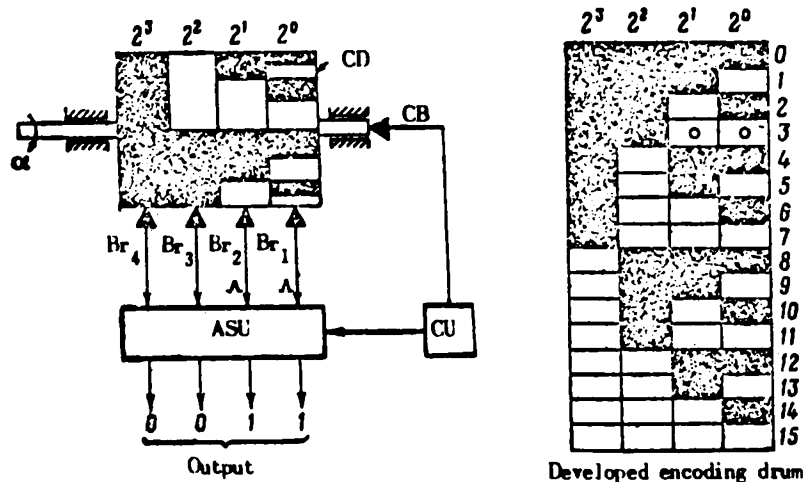


Fig.208 Shaft-Position Digitizer with Encoding Drum

in contact with conducting areas, pulses will arise simultaneously at several inputs of the ASU. After amplification and shaping, these pulses form the parallel code of the number at the output of the circuit.

This device operates as follows: When measuring the angle of rotation  $\alpha$  of the input shaft, a "query" pulse is sent from the CU to the CB, and causes the appearance of pulses at those inputs of the ASU to which brushes that happen to be in contact with the conducting areas of the drum are connected. At the positions of the brushes and various areas of the drum shown in Fig.208 (the brushes are denoted by circles on the drum development) pulses appear at the two right inputs of the ASU. The parallel pulse code of the binary number 0011, in this case, is obtained at the output of the converter.

A converter of the "shaft-digitizer" type, using an encoding drum, is a positional system that operates independently of the sense of rotation of the input shaft. To each position of the input shaft and therefore to each position of the code drum, there corresponds a certain binary number. To match the elements of the circuit, it is sufficient, for  $\alpha = 0$ , to let all brushes make contact with the nonconducting areas.

If shaft rotation angles are to be represented by binary numbers with many

places, one increases the number of tracks from which the codes are read, /356  
on the drum. In some cases the drums used may have a rather small number of columns but be coupled at certain transmission ratios. Figure 209 shows a symbolic diagram of a converter with four three-column encoding drums CD, coupled at a transmission ratio of 8 : 1. The angles  $\alpha$  in this converter are represented by the numbers from 0 to  $2^{12}$ .

Shaft-position digitizers with encoding disks are the most widely used of the direct coding systems, in view of the possibility of extensive use of the

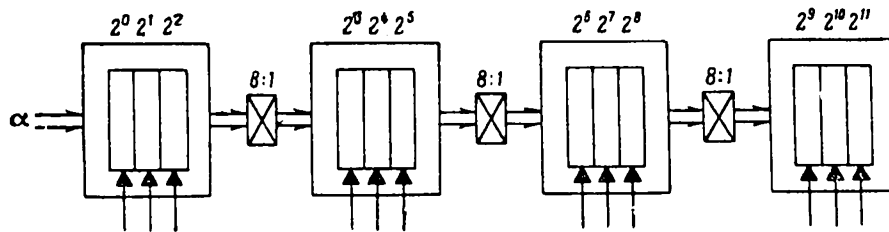


Fig.209 Shaft-Position Digitizer with Four Encoding Drums

photoelectric and inductive methods of data mapping. They are also compact and rather reliable. The method of data mapping has no effect on the design principles and only the design of the individual assemblies need be modified. Therefore, certain general problems will be considered below without reference to the method of data mapping.

Figure 210 is a schematic of a "shaft-digitizer" system with a four-column encoding disk and photoelectric data mapping. This system is based on the encoding disk CD (Fig.211), which consists of the concentric circumferences of a series of rings, one for each place in the binary numbers obtained by the conversion. The outer ring of the disk corresponds to the least significant digit of the numbers, and the inner ring to the most significant digit. Each ring is divided into a series of transparent and opaque areas of the same size. If the rings are numbered consecutively from the inner ring, then the number of transparent and opaque areas of any ring of the disk can be calculated by the formula

$$M_k = N_k = 2^{k-1},$$

where  $k$  is the number of the ring;

$M_k$  is the number of transparent areas of the  $k$ -th ring;

$N_k$  is the number of opaque areas of the  $k$ -th ring.

A transparent area of the ring corresponds to the code 1, an opaque area to the code 0. Successive transition from sector 0 to sector 15 therefore /357  
will yield a series of 16 successive binary numbers from 0000 to 1111. For example, the number 0110 corresponds to sector 6, the number 0100 to sector 8, the number 1010 to sector 10, and so on.

The encoding disk is rotated between the light sources LS and the photo-cells Ph (see Fig.210), arranged along the radius of the disk. Narrow light beams from the LS pass through the points of the axial circumferences of the

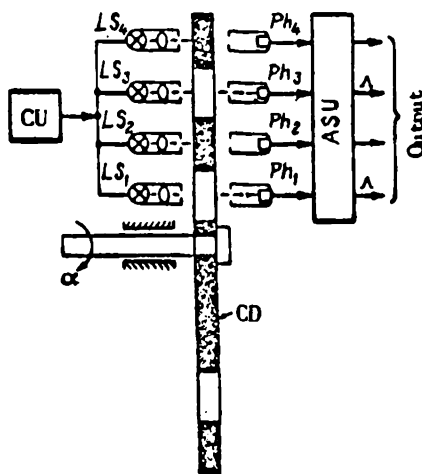


Fig.210 Shaft-Position Digitizer with Encoding Disk

corresponding rings of the disk, for example along the arc AA (see Fig.211); the light beams are interrupted by the opaque areas or not interrupted by the transparent areas of the disk, depending on its position, i.e., on the angle of

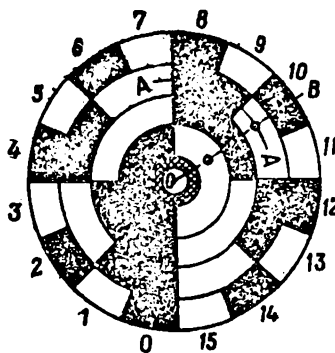


Fig.211 Four-Column Encoding Disk

rotation  $\alpha$  of the input shaft.

The bulbs of the light sources are lit only at the instant of measuring the angle  $\alpha$ , owing to the arrival of a special control pulse from the control unit CU.

The light beams from the LS only reach those photocells that, at the given

instant, are not covered by opaque areas of the disk. At the outputs of the excited photocells, and thus also at the corresponding outputs of the amplifier-shaper ASU, the parallel pulse code of the binary number, forming the discrete equivalent of the angle  $\alpha$  of the input shaft rotation, is produced.

In the position of the disk shown in Fig.210, the light beams pass from  $LS_1$  and  $LS_2$  to  $Ph_1$  and  $Ph_2$ , respectively. The parallel pulse code of the binary number 1010 is therefore formed at the output of the system. In this case, sector 10 of the disk is between the light sources and the photocells, as shown in Fig.211 by the location of the circles representing the photocells, along the radius of CB.

This device is a positional system, since any angle of shaft rotation is uniquely determined by the corresponding binary number.

If the inductive system of data mapping is used in such a device, then /358  
a pair of magnetic cores with windings is substituted for each pair of light source and photocell. One core acts as the signal transmitter, the other as

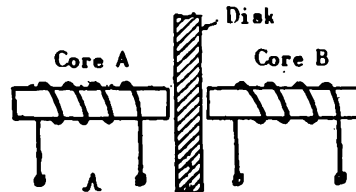


Fig.212 Magnetic-Core Device for Inductive Data Mapping

the receiver. The cores are so positioned that one of them forms the prolongation of the other (Fig.212). They may be  $\Pi$ -shaped.

Let there be only an air gap between the cores. When a current pulse is then fed to the winding of one core, an emf will be induced in the winding of the other core, and a current pulse will be produced at the output. This is due to the fact that the magnetic flux arising in the first core is closed across the second core, intersecting the turns of its winding. If, when a current pulse is fed to the winding of core A, a copper disk (screen) is placed between it and the core B, then no emf is induced in the winding of the core B and no pulse will appear at the output, since the magnetic flux arising in the core A is not closed across the core B but loses its energy by the formation of Foucault currents in the screen. To obtain the maximum screening effect, the screen is made of materials of low electric resistivity, for example red copper.

Thus, the principal elements of the converting portion of a "shaft-position digitizer" system with an inductive system of data mapping are magnetic cores with windings and a magnetic encoding disk with openings pierced in accordance with the code of the numbers to be obtained at the output. A comparison of the encoding disks of the inductive and photoelectric systems of data mapping shows



that the apertures in the encoding disk in the inductive system correspond to the transparent areas of the disk in the photoelectric system, and the continuous part of the disk in the inductive system to the opaque parts of the disk in the photoelectric system.

The encoding disk can be subdivided into areas according to various codes. Thus the disk shown in Fig.211 is subdivided into areas according to the conventional binary code. The development of such a disk or, expressed differently, the development of the ordinary binary code is shown in Fig.213a, where the

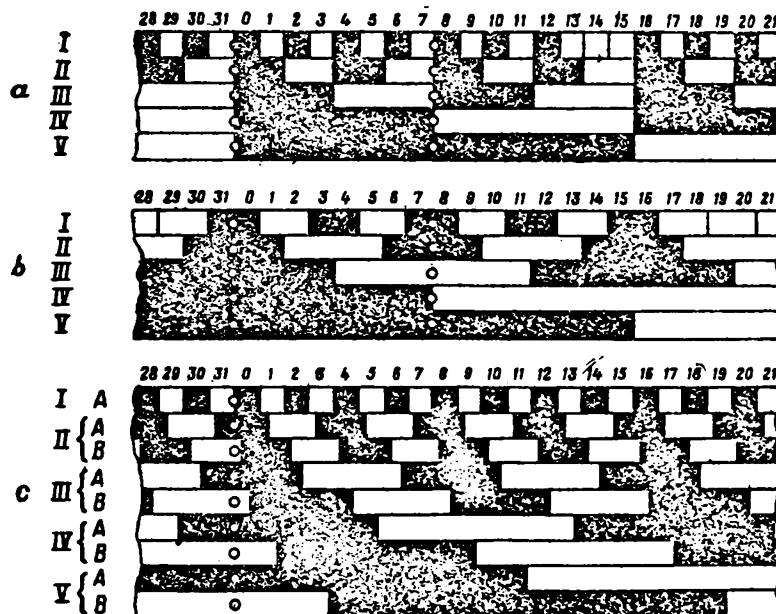


Fig.213 Development of Codes  
a - Binary; b - Cyclic; c - Shifted binary

dark portions correspond to 0 and the light portions to 1.

The ordinary binary code has the advantage that the parallel (or serial) code of the binary number can be obtained directly at the output of the unit, without any conversion. Large errors, however, may occur in surveying the data. In fact, in the ordinary binary code, two adjacent numbers may differ in all their digits. For this reason, in partitioning the disk into areas, the two sectors representing these numbers will differ in all their areas. Thus, on the boundary between sectors 31 and 0, or between sectors 15 and 16 (see 1359 Fig.213a), the dark areas in all places are replaced by light areas and vice versa. Since the sensitive elements (magnetic cores or photocells) are always positioned with a certain error and have a limited resolving power, either code 1 or code 0 can be read at the boundary between two areas of the disk. Therefore, at the boundary between sectors 31 and 0, for example, the code of any number from 00000 to 11111 can be read.

The reading errors with the usual binary code can be eliminated by stopping the disk at the instant of reading in a position such that the pickups cannot lie on the dividing line between adjacent sectors. But the use of special fixing components would complicate the design and sharply lower the speed of the entire system. For this reason, in high-speed shaft digitizers, where the data are taken off at high frequency, the disk is subdivided according to special codes in order to eliminate the reading errors. Such codes are: cyclic or reflex, and the binary-shifted, or V-scan code. Figure 213b is a 360 development of a cyclic (reflected) code and Fig. 213c of a binary-shifted (V-code).

The cyclic or reflected code is obtained from the ordinary binary code by substituting its digits according to a definite law. The representations of two adjacent numbers in the cyclic code differ from each other in the value of only one of the digits.

The cyclic code uses the same symbols for zero and one as in the ordinary binary code, i.e., 0 and 1. For two and three, the symbols 0 and 1 are written in the reverse order, and to distinguish their representations from those of 0 and 1 in the second place, the symbol 1 is recorded. Thus zero, one, two, and three in the cyclic code are represented by 00, 01, 11, 10. For the numbers from four to seven, these symbols are repeated in inverse order, but with the symbol 1 in the third place: 110, 111, 101, 100. The symbols for the following numbers are similarly obtained.

TABLE 24  
NOTATION OF NUMBERS IN BINARY AND CYCLIC CODES

Number	Binary Code	Cyclic Code	Number	Binary Code	Cyclic Code
Zero	0000	0000	Eight	1000	1100
One	0001	0001	Nine	1001	1101
Two	0010	0011	Ten	1010	1111
Three	0011	0010	Eleven	1011	1110
Four	0100	0110	Twelve	1100	1010
Five	0101	0111	Thirteen	1101	1011
Six	0110	0101	Fourteen	1110	1001
Seven	0111	0100	Fifteen	1111	1000

Table 24 gives examples for the notation of the sequence of numbers from zero to fifteen in the cyclic code, with the representation of these numbers in ordinary binary code for comparison.

It will be seen from Table 24 that the sequence of number representations in this code consists of cycles in each place, the digit sequences in any cycle being a mirror image of the digits in the next cycle of that place. In the

first place of the number representations in cyclic code, we note cycles of two digits each, in the second place of four digits each, and so on. If we represent the mirror images by horizontal lines, we obtain the following breakdown, by cycles, of the digits of the three least significant places of the 361 numbers presented:

0	0	0
0	0	$\frac{1}{1}$
0	1	$\frac{1}{1}$
0	1	$\frac{0}{0}$
1	$\frac{1}{1}$	$\frac{0}{0}$
1	1	$\frac{1}{1}$
1	0	$\frac{1}{1}$
$\frac{1}{1}$	0	$\frac{0}{0}$
$\frac{1}{1}$	$\frac{0}{0}$	$\frac{0}{0}$
1	0	$\frac{1}{1}$
1	1	$\frac{1}{1}$
1	1	$\frac{0}{0}$
0	$\frac{1}{1}$	$\frac{0}{0}$
0	1	$\frac{1}{1}$
0	0	$\frac{1}{1}$
0	0	0

These features of this particular code are responsible for its name, namely, cyclic or reflex (reflected) code.

The general rule of formation of the cyclic code from the ordinary binary code is as follows: To form the cyclic code of a number, its ordinary binary code must be shifted one place to the right and the shifted and unshifted ordinary binary code must be added, without carries from column to column, neglecting the least significant digit of the shifted code.

Thus, given the binary number

$$A_2 = a_{n-1}a_{n-2} \dots a_{i+1}a_i a_{i-1} \dots a_2 a_1 a_0,$$

its representation in reflected code

$$A_c = a_{n-1}\alpha_{n-2} \dots \alpha_{i+1}\alpha_i \alpha_{i-1} \dots \alpha_2 \alpha_1 \alpha_0$$

is obtained by forming the values of the digits  $\alpha_i$  of the latter according to the formula

$$\alpha_i = a_i + a_{i+1} \pmod{2},$$

i.e.,

$$\alpha_i = 0, \text{ if } a_i = a_{i+1}, \text{ and } \alpha_i = 1, \text{ if } a_i \neq a_{i+1}.$$

For example, let the number  $A_2 = 1101011$ . Then, its representation in cyclic code will be 1011110, since

$$\begin{array}{rcl}
& 1101011 \rightarrow & \text{unshifted code } A_2 \\
+ & 110101 \rightarrow & \text{shifted code } A_2 \\
\hline
& 1011110 \rightarrow & \text{cyclic code } A_c
\end{array}$$

The use of the cyclic code minimizes the reading error on the boundary /362 between two sectors of the encoding disk, which does not exceed a unit of the least significant column. This is explained by the peculiarities of the reflected code, in which two adjacent numbers differ in the value of only one of the digits, as will be clear from the Table and from Fig. 213b. Thus, the principal advantage of the cyclic code is that it minimizes the error of conversion of a continuous quantity into a discrete quantity.

However, the practical application of the cyclic code is limited by the difficulty of the conversion into ordinary binary code. If the encoding disk is subdivided into areas according to the cyclic code, then the numbers obtained at the output of a shaft-position digitizer will also be in cyclic code. For further utilization, the numbers must be represented in a code acceptable to the computer, i.e., one must change from the cyclic code to the ordinary binary code. It is the necessity of this transition that makes the design of actual circuits difficult. The use of cyclic code also complicates the design of two-channel converters.

The general rule for translating cyclic code to ordinary decimal code is as follows: Given the representation of a number in cyclic code:

$$A_c = \alpha_{n-1}\alpha_{n-2}\dots\alpha_{i+1}\alpha_i\alpha_{i-1}\dots\alpha_2\alpha_1\alpha_0,$$

then its representation in ordinary binary code

$$A = a_{n-1}a_{n-2}\dots a_{i+1}a_i a_{i-1}\dots a_2a_1a_0$$

is obtained by forming the bits  $a_i$  of that code according to the formula

$$a_i = \alpha_i + \alpha_{i+1} \pmod{2},$$

i.e.,

$$a_i = 0, \text{ if } \alpha_i = \alpha_{i+1}, \text{ and } a_i = 1, \text{ if } \alpha_i \neq \alpha_{i+1}.$$

Here, as in translating ordinary binary code to cyclic code, the most significant digit of the number representation always remains unchanged, i.e.,

$$a_{n-1} = \alpha_{n-1}.$$

A device for converting cyclic code to ordinary binary code may use various systems. The simplest is a system in which the logical operation of non-equivalence is performed, i.e., an OR-OR gate is used (Fig. 214).

The cyclic code of the number is fed, with the most significant digits

first, to the input A of an OR-OR gate. The codes formed at the output P of the circuit are fed to input B. The code of the first column, formed at the output P of the number, enters the input B of the OR-OR gate simultaneously /363 with the code of the second column of the number entering the input A, etc. Owing to this, the ordinary binary code of the number is formed at the output of the gate, likewise with the most significant digits first. To prevent the least significant digit of the number, which must not participate in the formation of a cyclic code, from passing through the OR-OR gate, the feed of the clock pulses  $I_c$  is interrupted at the required time. If this cannot be done, then a controllable AND circuit is connected to the output of the OR-OR gate, so that only the required number of code pulses of the number are passed at

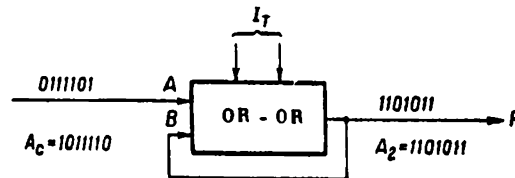


Fig.214 Schematic Diagram of Translation of Cyclic Code into Binary Code

the output of the converter.

The shortcoming of this system of translating cyclic code into ordinary binary code is that it operates on numbers which are transmitted on the code buses, with the most significant digits (MSD) first. Since operations on numbers are ordinarily performed by transmitting them in serial code, with the least significant digits (LSD) first, a device for modifying the sequence of the digits of the numbers on the code buses must be added to the system.

The binary-shifted code, or V-scan code is obtained by appropriate designing methods based on rules that differ from the rules for dividing the encoding disk into areas when the ordinary binary or cyclic code is used. The peculiarity of the binary-shifted code, namely, a double representation of each digit of the binary number, is expressed in the fact that two rings (two channels) are allotted on the encoding disk for the representation of a single digit. Accordingly, two pickups are required for each place, and the values of the codes are taken off according to definite rules. The rules of constructing the binary-shifted code and the rules for reading the information when such a code is used are such that the errors of conversion of the shaft position into numbers are decreased to a unit of the least significant place. In this case the numbers at the system output are in ordinary binary code.

The construction of the binary-shifted code will be illustrated by the aid of Fig.215, which gives a development of this code. The rules of code construction are as follows:

1. To represent each place of a binary number, two channels or subareas A

and B are allotted.

2. In the subareas A, the dark areas, representing the code 0, are so shifted that at a certain initial line the dark area of channel A of the  $(i + 1)$ -th place is shifted to the left relative to the dark area of the channel A of the  $i$ -th place by a quantity equal to a quarter length of an area <sup>/36/</sup> of the  $i$ -th place. The boundary line of transition from the sector representing the number 0 to the sector representing the highest number ...1111 is usually taken as the initial line NN. The dark areas of the channels A at the line NN are shifted according to the rule given above: The distance between the sections MM and PP, for instance, is a quarter length of the area of the II place.

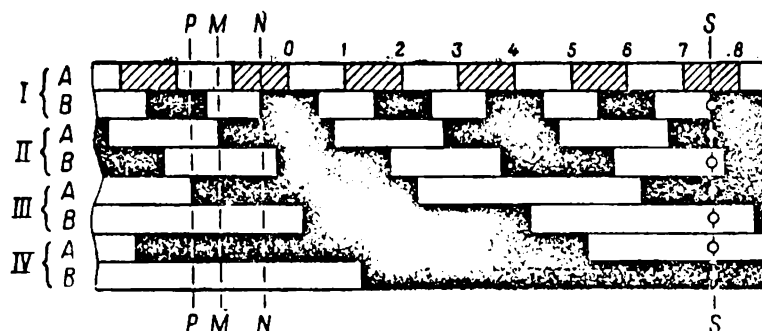


Fig.215 Development of Binary-Shifted Code

3. In the subareas B, all the dark and light areas representing the codes 0 and 1 are shifted relative to the areas of the subareas A to the right, by a quantity equal to half the length of an area in that place. The values of the light and dark areas of one place are the same as they are when ordinary binary code is used.

4. Channel A in place I is excluded from the development of the code, since when the binary-shifted code is used it is sufficient to have a single channel in the least significant place.

If the encoding disk of a shaft-position digitizer is subdivided into areas according to the binary-shifted code, then as many pickups as there are channels on the disk will be required to read the data. In the general case the angles of shaft rotation are converted into  $n$ -place binary numbers, which means that the converter must have  $2n-1$  pickups (two for each place, except the least significant place). All of them lie on a single straight line along the radius of the disk and are switched on according to definite rules.

The mapping of data from the pickups is controlled by the following rules when using the binary-shifted code:

If the code 0 occupies the  $i$ -th place, then the  $(i + 1)$ -th place is read from the pickup of the subarea A.

If the code 1 occupies the  $i$ -th place, then the  $(i + 1)$ -th place is read from the pickup of subarea B.

When these rules are followed, the conversion error for the shaft position into binary numbers is minimized, and does not exceed a unit of the least 365 significant place.

Consider the case of reading at the boundary between the sectors representing the numbers seven and eight. If the disk is subdivided in accordance with the ordinary binary code (cf. Fig.213a), then any number at all, from 0000 to 1111, can be read out. If the encoding disk is subdivided according to the binary-shifted code, either the number 0111 or the number 1000 will be read, depending on the position of the first-place pickup relative to the cross section SS of the development (Fig.215). If the pickup of place I is so located that the code 1 is read from it, then reading in the place II is done from the pickup of the subarea B, which also gives the code 1. In the place III, the code 1 is again read from the element of the subarea B. In the place IV, reading from the element of the subarea B gives the code 0. Thus, the code of the binary number 0111 is formed at the output of the system. If the pickup of the place I is so located that the code 0 is read from it, then the code of the binary number 1000 is formed at the output of the unit.

The use of the binary-shifted code results in two substantial shortcomings of the unit:

The encoding disk must be considerably larger, since two channels are reserved for each place.

The conversion of shaft position into  $n$ -place binary numbers requires  $2n-1$  pickups connected with a special switching system that permits the reading of information in accordance with definite rules.

The first of these two drawbacks is eliminated as follows: In dividing the encoding disk into areas, only a single channel (ring) is reserved for each place, while the pickups, whose total number still remains  $2n - 1$ , have the same location relative to the areas of the disk as they have when each place is represented by two channels. The channels of the subareas A are usually employed as the place channels, for which reason the pickups of these subareas are arranged in the same manner as when two channels are reserved for each place (Fig.216). Figure 216 shows the position of the pickups in reading the code 0 for the cases of two-channel and one-channel representation of the places in binary-shifted code.

The second drawback is less important since, when magnetic cores are used, the increase in the number of pickups does not particularly interfere with the design of the converter and, when the information is read in serial codes, the switching circuit is rather simple.

Figure 217 shows one version of the switching circuit, consisting of five ferrite-transistor cells. The circuit feeds the output of the converter with the codes read either in the subarea A or in the subarea B, according to the 367

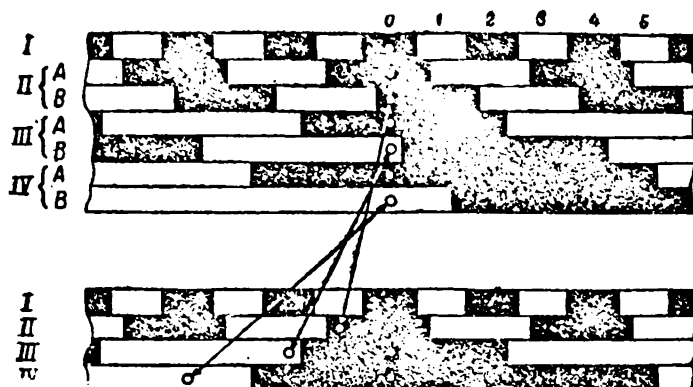


Fig.216 Formation of Binary-Shifted Code by Displacing the Pickups

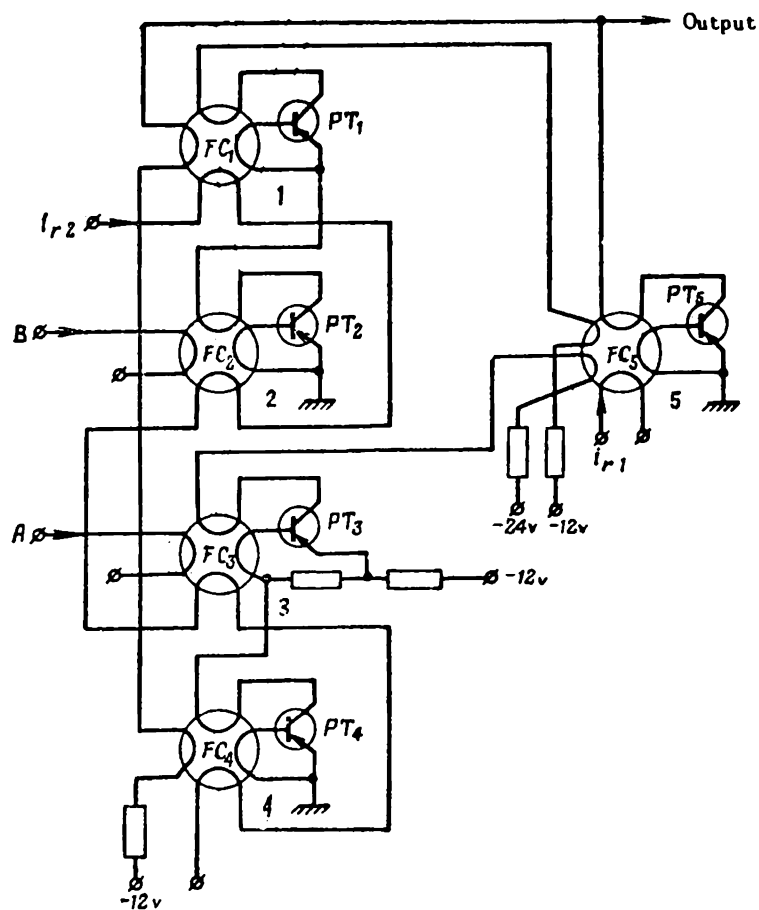


Fig.217 Switching System



above rules. This is accomplished by having the cells 1 and 2 form an AND gate, and the cells 3 and 4 an inhibit circuit, while the output signal formed at the output of cell 5 is fed to one of the inputs of these gates.

Figure 218 is a general schematic of a shaft-position digitizer with an encoding disk subdivided into areas in accordance with the binary-shifted code.

In this device, the term converter is applied to the encoding disk proper and the system of pickups used for reading the information. The delay line, which plays the role of a local control unit, maps the data in the form of serial pulse codes. If the data are required in the form of parallel codes, then the delay line is not used and the number of output amplifiers and switching circuits is increased according to the number of places on the disk.

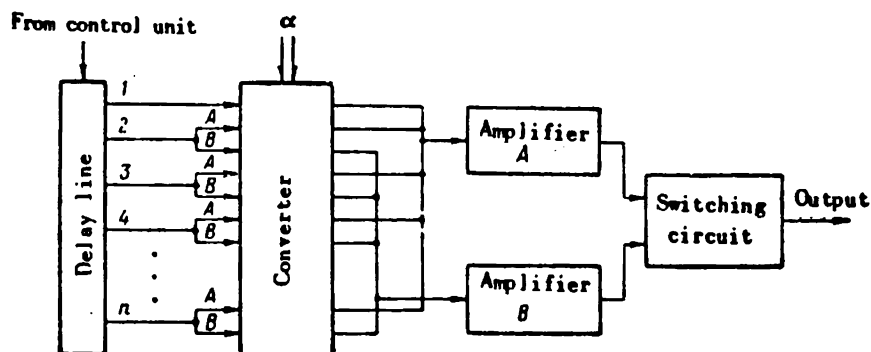


Fig.218 General Scheme of a Shaft-Position Digitizer with Encoding Disk

This system operates as follows: The beginning of the readout of the serial code of the number, into which the input shaft angle  $\alpha$  is converted, is determined by the time the command pulse is fed from the control unit CU to the input of the delay line. This pulse, which appears at the outputs of the delay line at each clock, successively "queries" the pickups of all places. The time required for passage of the pulse through the delay line is so selected that the pickups of all the places are "questioned" at a practically motionless encoding disk.

The code pulses are taken pairwise from the pickups, i.e., simultaneously from the subareas A and B, are fed after amplification to the input of the switching circuit, which selects them. As a result, the serial code of the number representing the discrete equivalent of the continuous magnitude of the input shaft angle  $\alpha$  is formed at the output of the unit. 1368

### Section 53. "Number-to-Voltage" Converters

Digital-to-analog converters for voltage are designed to effect the transition from binary numbers in parallel or serial code to the corresponding

voltages. Various principles and elements can be used to synthesize the circuits. Below, we will discuss an electronic number-to-voltage converter based

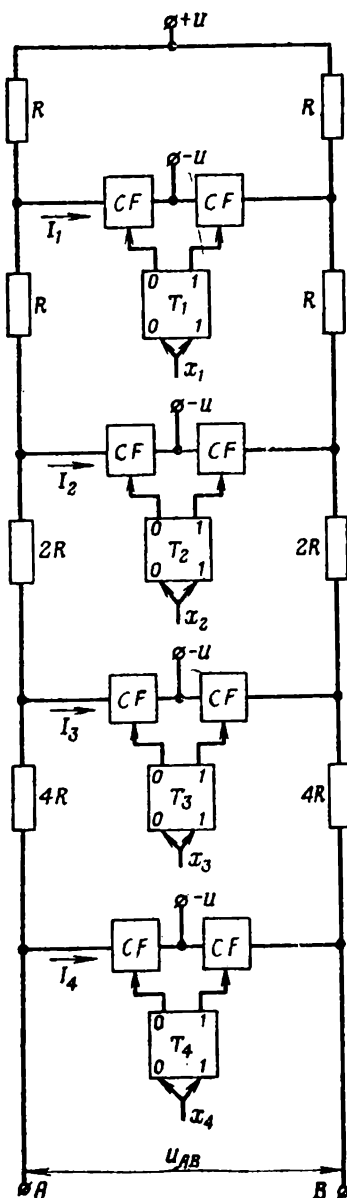


Fig.219 Schematic of System for Converting Binary Numbers into Voltages, with Current Stabilizer

on the principle of current addition and a converter based on ferrite-transistor cells.

A general schematic of a unit for conversion of four-place binary numbers

on the principle of current addition is shown in Fig.219. It consists of flip-flops forming the register of the number to be converted  $\bar{X} = x_4x_3x_2x_1$ , cathode followers CF to stabilize the currents, and two networks of progressively increasing resistors. The output voltage of the unit is denoted by  $U_{AB}$ .

The cathode followers not only have the function of current stabilization but also the function of switching the currents according to the state of the driving flip-flops. If the flip-flop is in the state 0, current will flow through the left-hand cathode follower of the pair connected to it, whereas if the flip-flop is in the state 1, current will flow through the right-hand cathode follower.

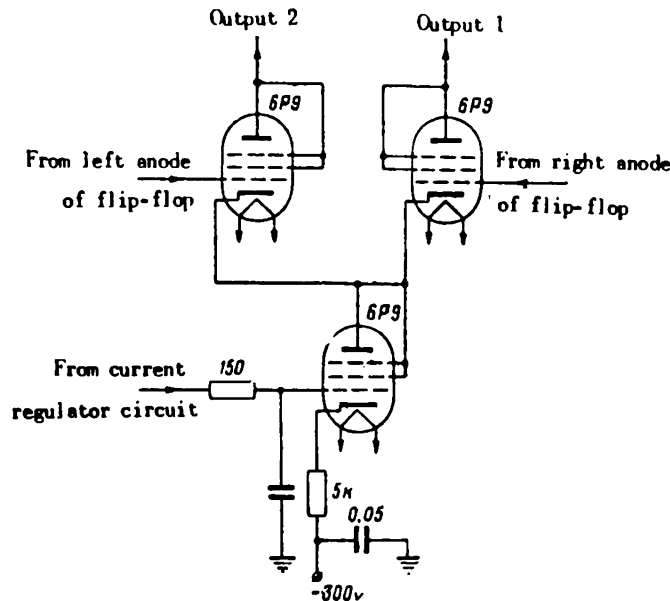


Fig.220 Circuit Diagram of Cathode Followers

The principal wiring diagram of a pair of cathode followers driven by one flip-flop is given in Fig.220. The common-cathode (common-emitter) circuit 369 of the followers has still another cathode follower instead of resistors, thus improving current stabilization.

When all flip-flops are in the state 0, i.e., when current flows through the left-hand cathode followers, the potential of point A is determined by the formula

$$U_A = U - (I_1 R + I_2 \cdot 2R + I_3 \cdot 4R + I_4 \cdot 8R) = U - 15\Delta U,$$

where

$$\Delta U = IR \text{ and } I_1 = I_2 = I_3 = I_4 = I.$$

Obviously, the potential point B is

$$U_B = U,$$

and the output voltage will be

$$U_{AB} = U_A - U_B = -15\Delta U,$$

If all flip-flops are in the state 1, i.e., if current flows through the right-hand cathode followers, the potentials of the points A and B and the output voltage will have the following values:

$$\begin{aligned} U_A &= U, \\ U_B &= U - 15\Delta U, \\ U_{AB} &= 15\Delta U. \end{aligned}$$

In the general case, when current can flow through the cathode followers in various combinations, the potentials of the points A and B are expressed 1370 as follows:

$$\begin{aligned} U_A &= U - 15\Delta U + p\Delta U, \\ U_B &= U - p\Delta U, \end{aligned}$$

where  $p$  is the decimal number corresponding to the binary number being converted, and introduced into the flip-flop register of the unit.

Then, the output voltage will be

$$U_{AB} = -15\Delta U + 2p\Delta U.$$

This expression shows that the voltage at the output of the unit is in fact proportional to the number being converted.

Let the binary number  $X = x_4x_3x_2x_1 = 1011$  be introduced into the flip-flop register of the converter. Then, current will flow through the right-hand cathode followers of the flip-flops  $T_1$ ,  $T_2$ ,  $T_4$  and the left-hand cathode follower of the flip-flop  $T_3$  (see Fig. 219). The potentials of the points A and B will then become

$$\begin{aligned} U_A &= U - 4\Delta U, \\ U_B &= U - 11\Delta U, \end{aligned}$$

and the voltage at the output

$$U_{AB} = 7\Delta U = -15\Delta U + 2 \cdot 11\Delta U,$$

thus confirming the validity of the above expression for the general case of conversion of a digital number into a voltage.

This system can also be designed to convert numbers with numerous digits. However, any increase in the number of digits will also increase the conversion errors.

Figure 221 is a schematic of a digital-to-analog converter based on ferrite-transistor cells. This unit consists of the number register P1, which has seven ferrite-transistor cells, four controllable AND gates, four generating cells GC, the pulse transformer PT, and an integrating RC circuit. It is designed to convert four-digit binary numbers into serial pulse codes.

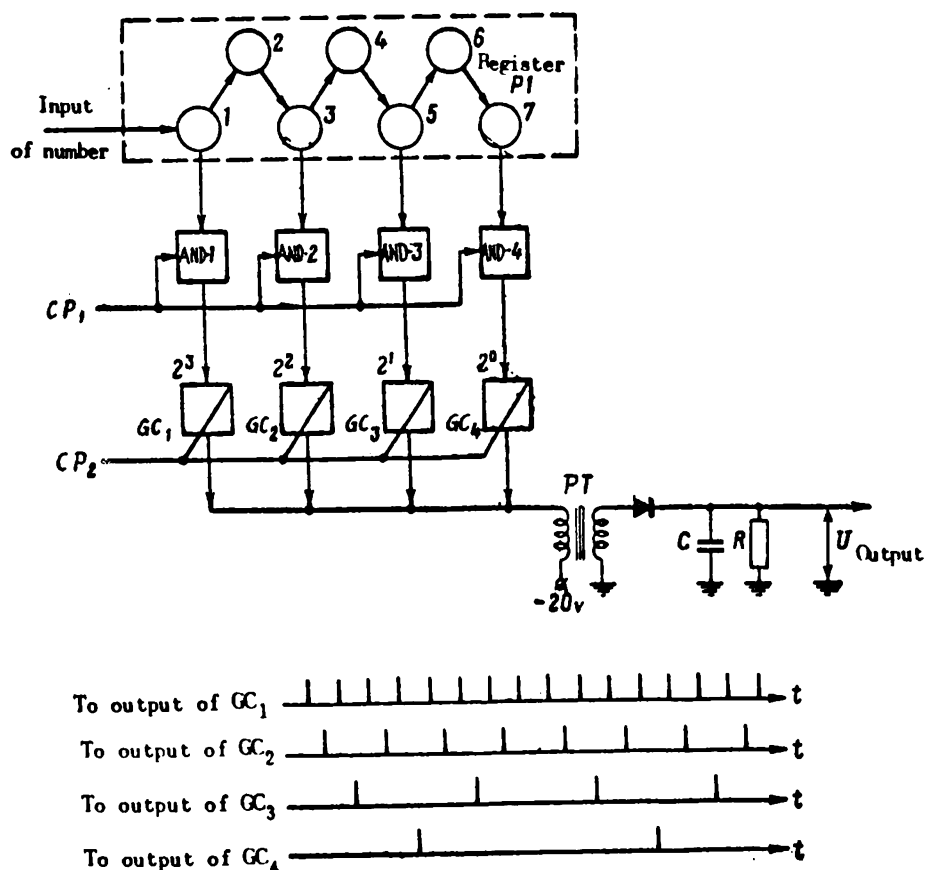


Fig.221 Ferrite-Transistor Circuit of a Digital-to-Analog Converter

The register P1 is used to record the number to be converted and then to transmit it in parallel pulse code through the controllable AND gates to the generating cells. The number is fed to the register, with the LSD first; therefore, when the register is filled, the least significant digit of the number will be written in the cell 7, and the most significant digit in the cell 1.

The generating cells  $GC_1 - GC_4$ , excited by the code pulses of the number

being converted, produce in unit time a number of pulses proportional to the digit in the corresponding place of the number. If the cell  $GC_1$  generates pulses of a frequency  $F$ , then the cell  $GC_2$  will generate pulses at a frequency  $F/2$ , the cell  $GC_3$  at a frequency  $F/4$ , and the cell  $GC_4$  at a frequency  $F/8$ . This frequency ratio is obtained by feeding the generating cells with clock pulses from a special frequency divider. /371

The outputs of all generating cells are connected to a common bus. The primary winding of the pulse transformer is also connected to this bus. The pulses of different frequencies formed at the outputs of the cells do not coincide in time, owing to their mutual shift, as will be seen from the timing chart at the bottom of Fig.221.

The integrating RC circuit separates the DC component of the pulse voltage, proportional to the number of pulses arriving in the primary winding of the pulse transformer PT. The time constant of the circuit is taken large enough to obtain, at the output of the unit, a voltage that can, for example, be used to drive the motor of a servosystem. /372

In this unit, a number is converted into the proportional voltage after formation of a pulse train whose number is proportional to the value of the number to be converted and depends on the frequency of pulse generation by the cells  $GC_1 - GC_4$  and on the operating time of these cells. The operating time of the generating cells is limited by the arrival of the control pulses  $CP_1$  and  $CP_2$  from the control block; it is calculated from the conditions for securing the required conversion frequency and may be only 20 - 50 msec.

Let the serial code of the binary number 1010 be introduced, for conversion into the proportional voltage, into the register P1, whose clock circuits are continuously fed with shift pulses. When the register P1 is filled, a control pulse  $CP_1$  is fed to the AND gate, and under the action of this pulse the parallel code of the number will arrive at the inputs of the generating cells. In our case, the code pulses 1 formed at the outputs of the cells 1 and 5, after passing through the AND-1 and AND-3 gates, will excite the generating cells  $GC_1$  and  $GC_3$ . These cells will generate until the control pulse  $CP_2$  is applied. During this time, the primary of the transformer will receive a pulse train whose number will be

$$N = M(2^3 + 2^1) = M \cdot 1010,$$

where  $M$  is a scale factor.

Since the output voltage is proportional to the number of pulses arriving at the primary of the transformer, it will also be proportional to the number converted.

The ferrite-transistor circuit of a digital-to-analog converter for voltage can also be so designed that all the generating cells will generate pulses of the same frequency, but feed them to different primaries of the pulse transformer, so that the signals are counted proportionally to the power of two, i.e., at a ratio  $2^3 : 2^2 : 2^1 : 2^0$ .

## Section 54. Number-to-Shaft Converters

Digital-to-analog converters for shaft position are designed to transform binary numbers, fed in some code, into the corresponding shaft-rotation angles. Such a converter may have either a closed-loop or an open-loop control system. If the converter has a closed-loop system, into which information is fed from output to input for comparison with information fed from outside the system, 1373 it is usually called a digital servosystem.

The most characteristic digital-to-shaft converters are those with step motors, those designed on the time-system principle, and digital servosystems.

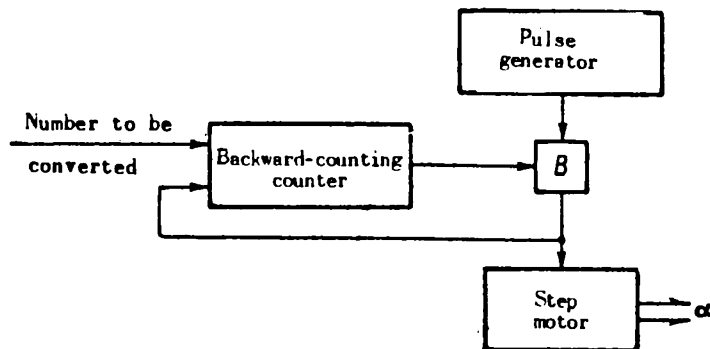


Fig.222 Digital-to-Analog Converter for Shaft Position  
with Step Motor

Converter with step motor. A number is most simply converted into a proportional angle of shaft rotation in a converter with a step motor, of which Fig.222 is a schematic. The converter consists of a pulse generator, a reversible counter, a gate B, and a step motor.

The backward-counting counter is used to compare the number being converted with the number of pulses arriving at the input of the step motor. It directly drives the gate B, in such a manner that the gate is blocked only if all flip-flops of the counter are in state 0.

The step motor plays the role of a final output element. On the arrival of each pulse, its rotor and thus also the output shaft of the system rotates through a definite angle, for instance 1/50 of a revolution.

This unit operates as follows: Pulses in a number proportional to the number to be converted are fed to one of the inputs of the reversible counter; in the limiting case this pulse train may represent the number-pulse code of the number to be converted. As soon as one of the flip-flops of the counter is set to the state 1, the gate B is opened and pulses from the generator begin to arrive at the step motor.

Each of these pulses arrives simultaneously at the second input of the

reversible counter. The number of pulses fed to the step motor therefore equals the number of pulses fed to the first input of the reversible counter, i.e., the number of pulses representing the binary number to be converted. Thus, the angle  $\alpha$  of rotation of the output shafts of the converter is proportional to the operand. /374

The conversion rate in this system is completely determined by the rate of pulse generation by the step motor and, consequently, cannot be high. Usually a step motor of a digital-to-shaft converter generates pulses at a repetition rate up to 15 cps.

Time systems. In time systems for conversion of a number to the proportional angle of rotation of an output shaft, the number is first converted into

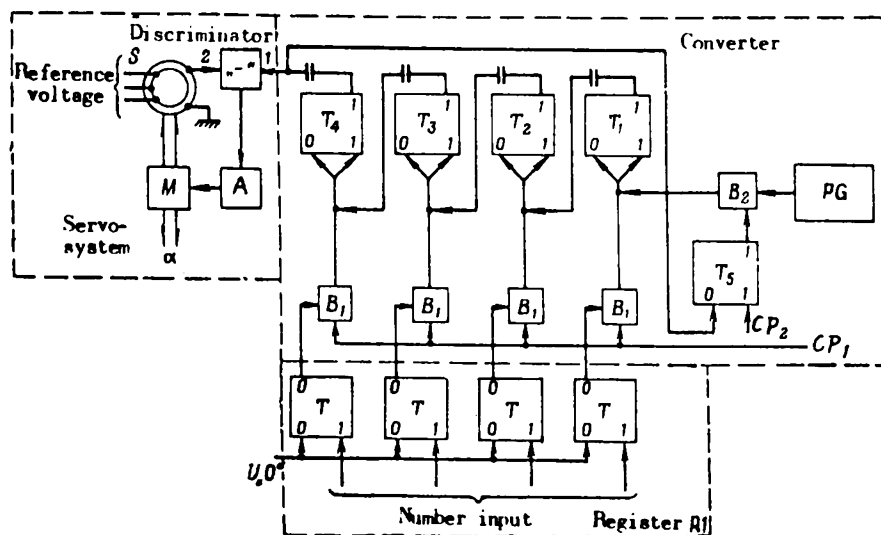


Fig.223 Digital-to-Shaft Converter with Servosystem

a time interval defined by the generation time of two pulses. This is due to the fact that servosystems capable of handling quantities assigned in the form of the corresponding time intervals are used here.

Figure 223 is a schematic diagram of a digital-to-shaft converter using the time-system principle. The unit consists of two main assemblies, a converter of the original number into a proportional time interval, and a servosystem. The diagram shows the case of the conversion of four-digit bits but it may also be used in other cases of number conversion with more places. The diagram also shows the number register Pl, which is ordinarily a part of some other unit of the computer.

The converter of the original number into a proportional time interval consists of a series of flip-flops and gates and includes a calibration oscillator PG. The number for conversion is fed in inverse code through the group of gates  $B_1$  from the register Pl to the register-counter of the converter, /37



consisting of the flip-flops  $T_1 - T_4$ . The gate  $B_2$  and the flip-flop  $T_5$  control the operation of the converter.

The device operates as follows: The binary number to be converted is first rewritten in the inverse code from the register P1 into the register of the converter. For this purpose, a control pulse  $CP_1$  is fed to the group of gates  $B_1$ , which are opened when the corresponding flip-flops of the register P1 are in the state 0. After setting up the inverse code of the number in the

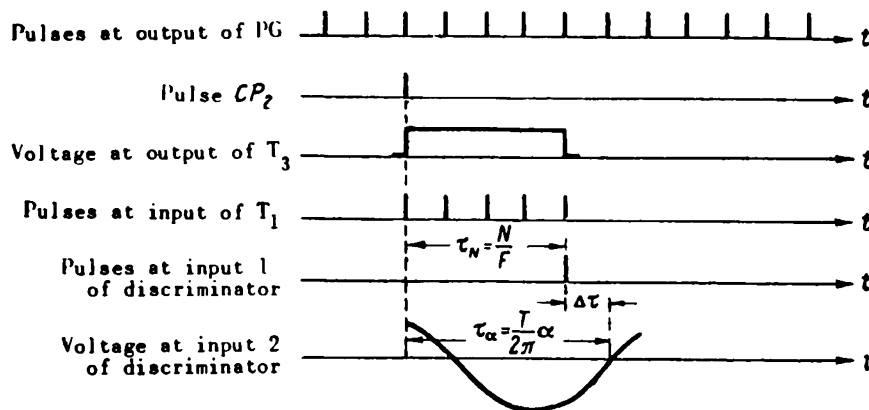


Fig.224 Timing Chart for One-Channel Digital-to-Shaft Converter

register of the converter, a control pulse  $CP_2$  is fed to the input of the flip-flop  $T_5$ , setting it in the state 1, at which the gate  $B_2$  is opened.

The time of feeding the control pulse  $CP_2$  determines the beginning of the time interval into which the original number is converted. Beginning at this instant, pulses are fed through the gate  $B_2$  to the input of the flip-flop  $T_1$  of the register-counter of the converter from the generator PG. The number of these pulses is proportional to the number to be converted and thus also to the corresponding time interval (Fig.224).

Gate  $B_2$  remains open until the register-counter is filled, i.e., until the flip-flop  $T_4$  is reset from the state 1 to the state 0. The pulse formed at the output of the filled register-counter resets the flip-flop  $T_5$  to the state 0, thus cutting off the gate  $B_2$  and stopping the feeding of pulses from the PG to the input of the register-counter. The time of formation of this pulse corresponds to the end of the time interval into which the original number is converted.

Consider the example of the conversion of a four-digit binary number 1376 into the proportional time interval (see Fig.223). Let us convert the binary number 0101. When its inverse code is fed to the register-counter, the number 1010 will be set up there. From the generator PG, pulses begin to arrive at the input of the register-counter from the instant the control pulse  $CP_2$  is fed to the flip-flop  $T_5$ . In this case, the register-counter will be filled

after the arrival of the sixth pulse from the PG. Thus, a pulse is formed at the output of the flip-flop  $T_4$  when the sixth pulse, denoting the end of the time interval, is fed to the input of the register-counter. This means that the resultant time interval contains five time segments, each equal to the pulse repetition rate of the reference frequency  $F$ , i.e.,

$$\tau = \frac{1}{F} 0101,$$

where  $\tau$  is the time interval.

In the general case, some number  $N$  is converted. Its proportional time interval will be

$$\tau_N = \frac{1}{F} N.$$

The pulse at the converter output, which appears at time intervals  $\tau_N$  after the control pulse  $CP_2$ , is fed to the first input of the discriminator of the servosystem. The pulse  $CP_2$  is a reference pulse; it is formed at the instant the sinusoidal reference voltage passes from minus to plus as is the case in phase-shifter time systems of shaft digitizers. The sinusoidal voltage from the selsyn  $S$  is fed to the second input of the discriminator; it is usually shifted a certain quantity  $\tau_\alpha$  in time, relative to the reference voltage (see Fig. 224).

If the values of  $\tau_N$  and  $\tau_\alpha$  do not coincide, an error signal  $\Delta\tau$  is generated and acts across the amplifier  $A$  on the motor  $M$ , turning the rotor of the selsyn  $S$  through the angle necessary to make  $\tau_\alpha$  practically equal to  $\tau_N$ . Consequently, the second stage of conversion of a binary number into the proportional shaft angle consists in a comparison of two time values and the reduction of their difference to zero.

The phase of the sinusoidal voltage fed to the second input of the discriminator is directly correlated with the angle of rotation of the selsyn rotor, and thus also with the angle  $\alpha$  of rotation of the output shaft of the unit. For this reason, when  $\tau_N$  and  $\tau_\alpha$  are equal, the angle  $\alpha$  of rotation of the output shaft will be proportional to the number being converted.

The quantity  $\tau_\alpha$  is expressed as follows:

237

$$\tau_\alpha = \frac{T}{2\pi} \alpha,$$

where  $T$  is the period of the sinusoidal reference voltage.

For  $\tau_N = \tau_\alpha$ ,

$$\frac{1}{F} N = \frac{T}{2\pi} \alpha,$$

i.e.,

$$\alpha = \frac{2\pi}{TF} N.$$

If larger numbers must be converted at high accuracy by a digital-to-shaft converter using the time system, it may be designed as a two-channel converter. In this case, the converter must have two transducers and a servo-system with two selsyns and two discriminators. The more significant digits of the number are converted on the channel GO, and the less significant digits on

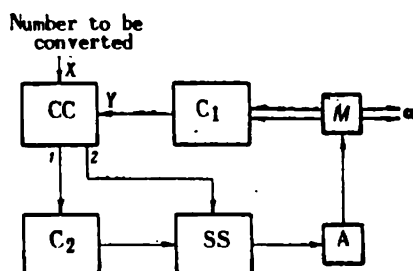


Fig.225 Block Diagram of Digital Servosystem

the channel TO, the number usually being handled on the channel TO, while the channel GO comes into operation only at large mismatch, i.e., at large values of  $\Delta\tau_{GO}$ .

**Digital servosystems.** A digital servosystem used for converting a binary number into the proportional angle of shaft rotation is a rather complex system, and contains other converters of discrete quantities into continuous quantities, and of continuous quantities into discrete quantities. Figure 225 gives a general block diagram of such a system. The unit includes a comparison circuit CC, the converters  $C_1$  and  $C_2$ , the switching system SS, the amplifier A, and the motor M.

The comparison circuit, which is the null element of a digital servosystem, compares the number X to be converted or handled with the resultant number Y coming from the converter  $C_1$ . The produced number Y means the number which, at the given instant, is proportional to the angle  $\alpha$  of rotation of the output shaft. At the output 1 of the comparison circuit a quantity proportional to the difference between the input and output numbers is formed and is fed in the form of some pulse code to the converter  $C_2$ . The coincidence signal for these numbers appears at the output 2 in the form of a control signal to the switching system SS.

The converter  $C_1$  is a shaft digitizer with encoding drums or disks. If the digital servosystem is designed to convert multi-column binary numbers, 378 then the converter  $C_1$  is built with two channels.

The converter  $C_2$  is a digital-to-analog converter, built with the same elements as the other assemblies of the system. Thus, in systems that make extensive use of ferrite-transistor cells, the converter  $C_2$  is usually based on a ferrite-transistor circuit. A voltage proportional to the difference between the numbers X and Y or to the imbalance of the numbers is obtained at the output of  $P_2$ . This voltage is used to control the output motor M. In many digital

servosystems, the proportionality of the output voltage of the converter  $C_2$  to the imbalance of the numbers is maintained only up to some definite degree of imbalance. When the imbalance increases beyond this, the voltage will remain constant, making it possible to handle more extensive mismatch between the position of the output shaft and the number being converted, at a certain speed which is maximum for that system.

The switching system is designed to reverse the polarity of the voltage fed from the converter  $C_2$  to the amplifier A, in order to eliminate the mismatch over the shortest possible path. The polarity of the voltage is reversed only if a special control signal, generated in accordance with the relation of the numbers X and Y, is fed from the comparison circuit. If the converter  $C_2$  itself changes (switches) the polarity of the voltage on signals received from the comparison circuit, then no special switching system is required.

A digital servosystem operates as follows: The number X to be converted is fed in parallel or serial code to the comparison circuit. The code of the number Y, proportional to the angle  $\alpha$  at the given instant, is fed simultaneously from the converter  $C_1$  to the comparison circuit. If the numbers X and Y are not equal, then a quantity proportional to their difference is fed from the comparison circuit to the input of the converter  $C_2$ . A control voltage is formed at the output of  $C_2$  and is then fed at a definite polarity to the amplifier, forcing the motor to eliminate the mismatch between the number being converted and the shaft angle  $\alpha$ , i.e., the number Y. As soon as the motor has rotated the output shaft through an angle such that the number Y taken from the converter  $C_1$  becomes equal to the number to be converted X, the compensating shaft motion stops. In this case, the difference between the numbers equals zero, and nothing is fed to the input of the converter  $C_2$  and thus also nothing to the input of the amplifier.

If a digital servosystem is designed to handle a continuously varying quantity, expressed as a sequence of binary numbers, then it is necessary, for smooth compensation, to feed the numbers to the input of the comparison circuit at a frequency of 10 - 15 cps.

## THE CONTROL UNIT

Section 55. General Characteristics of the Control Unit

The control unit of an electron digital computer is designed for the automatic execution of a given program for the solution of a problem (problems) by means of the forced coordination of operation of the other units of the computer. The forced coordination of operation of the other computer units is effected by the control unit which emits pulse- or potential-type control signals that can be produced at a given instant of time, to the other units and components. Thus, the main function of the control unit is the conversion of the primary instruction information, represented by the problem-solving program, into secondary instruction information which can be represented by the control signals whose effect on the other computer units ensures automatic solution of the problem.

The problem-solving program is a sequence of conditional number instructions stored in the memory unit along with the source data. Each instruction determines the action of the computer with respect to the performance of some operation and constitutes an element of the primary instruction information. The instructions must be selected from the memory unit and executed in the order assigned by the overall computational program. To simplify the readout operations, the instructions, as a rule, are allocated in the memory-unit cells which are numbered consecutively. If the first instruction is allocated in the cell with the number (address) 0100, the second is allocated in cell No.0101, etc. When the instructions are so allocated, their addresses are determined automatically, by counting the pulses produced during each instruction readout.

In most cases, the instruction selected from the memory unit cannot be converted into a corresponding series of control signals during one cycle. Therefore, the control unit should ensure not only the conversion of the instruction, but also its storage for a specific time interval. The access time, the storage and conversion of one instruction into a corresponding series of 380 control signals, i.e., the execution of this instruction, is called the operation cycle of the control unit. For most computers, the operation cycle of the control unit coincides with the total operation cycle of the computer.

The digits of an instruction like those of any conditional number determining the performance of the computer in one cycle, are divided into two basic parts: operation and address. The operation part, or operation code, determines the performance of the computer units in a given cycle, i.e., determines what operation must be carried out. The address part of the instruction determines the number (address) of the cells of the memory unit which store the numbers with which the given operation must be effected.

The amount of addresses in the address part of an instruction is constant

for each computer, and may be equal to 1, 2, 3, and 4. It determines the number of accesses to the memory unit during one cycle. Thus, in the case of three-address instructions, it is necessary, during one cycle, to use the memory unit four times: to select the instruction, to select from the first instruction address the first number, and from the second address the second number with which the given operation will be effected and, finally, to forward from the third address the result of execution of the operation. Since, in addition to using the memory unit during the cycle time  $T_c$ , it is still necessary to use the numbers, it follows that, in the general case,

$$T_c > (k + 1) T_{mu},$$

where  $k$  is the number of addresses in the instruction, or the machine address;  $T_{mu}$  is the time of one access to the memory unit.

If the operation time of the arithmetic unit  $T_{au}$  with respect to the execution of a given instruction is less than the time of access to the memory unit, then coincidence of the operation cycles of the arithmetic and memory units may occur. Then,

$$T_c = (k + 1) T_{mu},$$

which points to the factual possibility of reducing the cycle time, i.e., the possibility of increasing the total high-speed performance of the computer.

The conversion of an instruction as an element of the primary instruction information must be examined separately for its address and operation parts, since the conversion of these parts does not occur uniformly. The conversion of the address part amounts to its separation into individual addresses which are the numbers of the corresponding memory unit cells, and to their transfer to the register address of the memory unit at given instants of time.

The operation part of an instruction should be converted into a corresponding series of control signals which immediately ensure the execution of one operation. The easiest way of effecting this conversion is as follows: The 381 operation code (the operation part of the instruction) is fed to the input of the decoder, on one of whose outputs a signal is produced which corresponds to the assigned operation. The unit generating the control signals, necessary for execution of one operation, is connected to each decoder output. Thus, when the operation part of the instruction is interpreted, a definite control signal generator is excited, which also ensures its necessary conversion. It should be noted that in many computers, the individual control signal generators are combined into one general unit controlling the execution of the operations (the operation control unit).

On the basis of the above statements, we can define the main functions of the control unit as follows:

- 1) Readout of the next instruction from the memory unit.

2) Storage of instruction during the cycle.

3) Conversion of the address part of the instruction: readout of the numbers participating in the operation from the memory unit and transferring the result to the memory unit;

4) Conversion of the operation part of the instruction: producing the control signals which provide for the execution of all operations specified by the computer circuit.

Moreover, the memory unit should ensure:

- a) input of the source data and, if necessary, insertion of the program;
- b) readout of the problem-solving results from the computer;
- c) starting and stopping the computer;
- d) control of operation of the computer;
- e) operation of the computer under various regimes (automatic, cyclic, etc.)

The layout of the control unit is determined by its basic and auxiliary functions as well as by the characteristics of the given computer. In order to read out the instructions from the memory unit, it is necessary to fix their consecutive numbers. This function of the control unit is readily carried out by means of a counter which is usually called a control counter. To store an instruction in the control unit during one cycle, a register is sufficient which consists of  $k + 1$  parts, i.e., which makes it possible to separate the addresses and the operation part of the instruction. This register, in combination with  $k + 1$  groups of gates, also ensures the conversion of the address part of the instruction. To convert the operation part of the instruction, decoders and control-signal units are used, consisting of groups of gates or designed on the principle of delay lines with a multiple number of outputs.

Thus, a control unit should consist of the following functional building blocks:

- 1) control counter;
- 2) instruction register;
- 3) operation-code decoder;
- 4) control-signal generator;
- 5) program input (initial starting unit);
- 6) control panel;
- 7) starting and stopping unit;
- 8) clock-pulse generator and certain other components.

/382

Control units are classified according to various characteristics. According to their capabilities, they are divided into universal control units and "hard" program units. The first ensure the execution of any program within the limits of possibilities of the computer; the second ensure the execution of individual fixed programs, for example, the repeated solution of the same problem when there are different source data in a special-purpose digital computer.

Depending on whether the cycle time is taken as constant or variable, the control units and the corresponding digital computers are divided into synchronous and asynchronous. In synchronous computers, execution of any instruction begins only some time  $T_c$  after the execution of the previous instruction has started, i.e., a predetermined time is set for the execution of any instruction. In asynchronous computers, execution of the next instruction starts immediately after completed execution of the previous instruction. Thus, synchronous computers have a constant cycle of operation while asynchronous computers have a variable cycle.

In computers with constant operation cycle, a part of the total time allocated for solution of the problem is spent in "idling", since most operations usually require much less time than the longest operation on which the cycle is laid out. Asynchronous control units provide for higher speed of the computer since there is no "idle time" in this case between the actual cycles of instruction execution. However, asynchronous control units are usually more complex than synchronous.

In order to give a control unit the advantages of both synchronous and asynchronous units, the operation of some computers, such as the BESM-2, is organized as follows: The basic cycle time is assumed constant and is determined from the duration of execution of most of the operations (addition, subtraction, comparison, etc.). These operations, whose execution time is not included in the assumed cycle (multiplication, division, and several others), are put into a separate group. The control of execution of the operations of this group is effected by a separate local operations-control unit whose operation time would be included in the basic cycle of the computer. Thus, the computer basically operates as a synchronous computer but, when necessary, the cycle time is increased by the operation time of the local operations-control unit.

Control units are further divided into centralized and hybrid. In the case of centralized control units, almost all or all of the control signals are generated by the central control unit; the other computer units in practice do not have local control units. Such units are used rather rarely; because of their complexity, they are suitable for small special-purpose computers. When hybrid control is used, the central control unit generates only the basic control /383 signals which are assigned for the local control units which immediately effect the control of operation of the corresponding units. Hybrid control is more flexible, which is its main advantage.

## Section 56. Control Units of Three-Address Computers

Three-address electron digital computers are usually designed like general-purpose computers, with great possibilities of solving problems of different classes. Their control units are generally universal with a constant basic cycle; moreover, their organization corresponds to the principles of hybrid control. The BESM-2 is a typical three-address computer.

The BESM-2 universal electron digital computer is of the 39-digit type. When an instruction is formed, this amount of binary digits is distributed in the following fashion: Six digits comprise the operation code, and the remain-



ing digits are divided into three 11-digit groups which are the addresses  $A_1$ ,  $A_2$ , and  $A_3$ . This arrangement of the addresses fully corresponds to the memory capacity of the computer, since  $2^{11} = 2048$ . The following is an example of an instruction for the BESM-2 computer:

Operation code	$A_1$	$A_2$	$A_3$
000001	10110001010	10001000100	10000001110

The sequence of realization of the code of such an instruction is usually as follows:

- preparation of the circuits necessary for execution of the operation whose code is indicated in the instruction;
- reception of one or two numbers participating in the given operation, from the working storage into the registers of the arithmetic unit;
- immediate performance of the operation;
- readout of the operation in the working storage;
- reception of the subsequent instruction.

Each operation is divided into elementary functions which are carried out in a fixed order. These functions may be: reception of the codes in the various units, readout of the codes to the code buses for their transfer to other units, addition of the two codes, shift of the codes to the right or left, setting the units or individual cells to the initial state, etc. To verify the correctness of these functions, the control unit of the computer can operate when the computer automatically stops after the execution of each elementary function. /384 Operation in cycles can also be used as a control method when the computer automatically stops after execution of each subsequent instruction. The basic regime is automatic when the computer operates without any operator before completion of the calculations from a pre-inserted program.

Figure 226 shows the block diagram of the control unit of the BESM-2 computer. This unit consists of the following components: the central control unit CCU; the instruction control unit ICU which consists of the instruction storage unit ISU, the central instruction control unit CIC and the local instruction control unit LIC; the operations-control unit OCU consisting of the operations commutator OC, the central operations-control unit COC, the local operations-control unit LOC and the interlock circuit IC; the control console or panel CP.

The central control unit CCU cyclically produces instruction pulses and potentials which are distributed to the control circuits of the computer, i.e., this central unit sets the operation time of the computer.

One cycle of the central control unit CCU corresponds to the execution of one instruction and the reception of the following instruction. During that time, two numbers are selected from the memory unit, a certain operation is carried out on them, the result is sent to the memory unit, and the next command is received.

If local control must participate in the execution of the next instruction,

then for the time of its operation, the central control stops and continues its operation only after completion of the operations of the local control. The time for execution of these operations consists of the time for passage of the cycle of the central control CCU and the time necessary for execution of the local control operation.

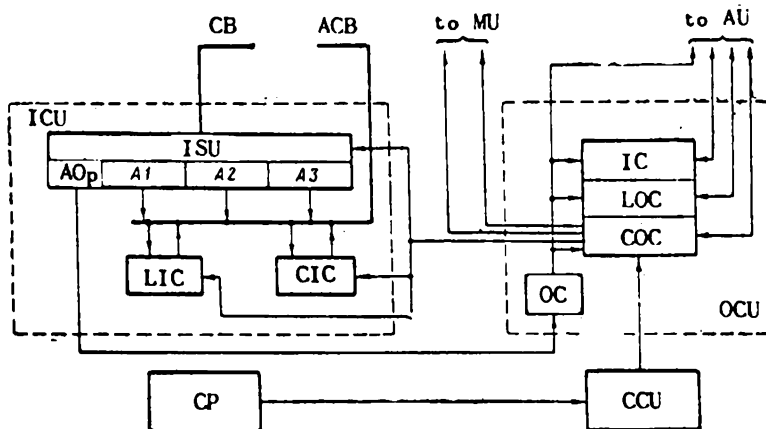


Fig.226 Control Unit of the BESM-2 Computer

The pulses and potentials generated by the CCU are directed to the central operations-control unit. The CCU itself operates from pulses coming from the /38 special main pulse generator MPG.

The instruction storage unit ISU is designed for the intermediate storage of the code of the instruction being executed. It consists of ordinary static flip-flops connected to the circuit of the register. Structurally, the instruction storage unit consists of four parts or flip-flop groups: the operation-code storage group AOp connected to the operations commutator; the first number address code storage group A1; the address code storage group of the second number A2; and the address code storage group of the result A3. The last three groups are connected by output gates to the address code buses ACB from which the address code of the number is transferred to the other units of the computer.

The instruction code is inserted in the instruction storage unit ISU from the working memory unit through the input gates which are connected to the code buses CB.

When a given computational program is carried out, the number of the instruction which must be taken from the working memory is determined by the code located in one of the instruction-control units (central CIC or local LIC).

The central and local instruction-control units CIC and LIC are used for the reception, storage, and generation of the instruction address codes, and for transferring these to the working storage control device. Both units are identical. Each of them is an 11-digit flip-flop counter having loops by which the counter is set to the code position "0"; this is done by setting all flip-flops

of the counter to the code position "1" and by subsequent feeding of the pulses to the input of the counter.

In most cases, the instructions of the program alternate in their number sequence: 00001, 00010, etc.; therefore, to determine the number of the following instruction, it is sufficient to feed the input of the instruction-control unit (CIC or LIC) with a single pulse which increases the value of the code stored in this functional block by unity. Sometimes (for example, during conditional transfer from the basic program to the subprogram and back), the number of the following instruction is given in the third address A3 of the previous instruction. Therefore, there are input gates in the CIC and LIC, which are connected to the address code buses ACB where the code of the third address is received from the instruction storage unit.

Both instruction-control units are used in the transfer from the basic program to the subprogram and back: One of them contains the number of that instruction of the basic program which must be carried out after completion of the calculations according to the subprogram; the instruction addresses of the subprogram are formed in the second.

The CIC and LIC units are also used to perform group operations on the numbers (transfer of groups of numbers from the external memory unit to the working memory and back, etc.).

Each instruction-control unit has output gates with cathode followers which are connected to the address code buses. The instruction number code is transferred through these gates at an appropriate instant of time, determined by 386 the operation of the central control unit CCU and of the central operations-control unit CCO, to the address code buses and is fed from there to the control element of the working memory, for readout of the required instruction.

The switching of the instruction control from central to local, i.e., the switching of the CIC and LIC units, is effected by the central operations-control unit as soon as the corresponding instruction arrives at the operations commutator.

The operations commutator OC converts the operation code, assigned by the number in the binary system of notation, into a control voltage at one of the 32 output buses and thereby prepares the loops of the other functional blocks for performance of this operation. The main portion of the operations commutator consists of a two-cascade diode decoder, whose input buses are connected across amplifiers to the outputs of the flip-flops of the AOp of the instruction-storage unit. Each of the 32 output buses of the decoder is connected to the external loops across a cathode follower, which makes it possible to obtain the necessary decoupling of the loops and increases the reliability of operation of the commutator.

The central operations-control unit COC controls the operation of the individual circuits of the computer during performance of the elementary operations which are common to many instructions and are included, with respect to time, in the cycle of the central control unit. Essentially, the COC consists of gates whose inputs are fed with the control voltages from the central control

unit. These voltages determine the moment of performance of any operation. The other inputs of the gates are fed with control voltages from the operations commutator. These voltages determine what operations should be performed upon a given instruction.

The control voltages of the gate outputs are fed across amplifiers or shaping units to the other units of the computer.

The local operations-control unit LOC controls the operation of the individual circuits of the computer during performance of elementary operations peculiar to certain instructions and requiring more time than the cycle of the CCU. The local operations-control unit LOC consists of a large number of circuits each of which controls one of the operations selected for local control (for example, normalization of the numbers to the left).

The control voltages are produced directly in the LOC since, in this case, the elements of the CCU and of the COC are disconnected from the other units of the control unit.

The interlock circuit IC discontinues the operation and changes the alternation of the elementary operations if any partial results occur in the arithmetic unit. This circuit is also used for automatic stop of the computer in the case of overflow of the column loop.

The control console CP is intended to control operation of the computer, to start and stop it, and to control its operation during the computations. On the console, there are toggle switches for manual, automatic, and semi-automatic operation, a start-and-stop button, a load button for the initial input of /387 the source data, control instruments and a mnemonic circuit with display lights which show the operation of the individual units and functional blocks of the computer.

Let us examine the operations of the units of the control unit during execution of a typical instruction sequence, when the address number of the subsequent instruction is one greater than the previous. Moreover, we will assume that the computational program and the necessary source data are inserted into the working memory unit.

Automatic operation of the computer can begin only after the first instruction code of the program has been placed in the instruction storage unit ISU. This instruction will be inserted in the ISU by the operator. The operator sets up the address code of the first instruction on the control console and forces its recording in the CIC unit. When the toggle switch on the console is set to "automatic operation", a pulse from the CCU registers the address code of the first instruction in the control unit of the working memory which brings about the readout of the code of the first instruction from the working storage and its write-in into the ISU.

As noted above, the instruction code in the BESM-2 computer usually consists of four groups which are the operation code and the number codes of the storage locations from which those numbers that participate in the operation

must be extracted or into which the result of the operation must be written. For example, the instruction code

<i>AO<sub>n</sub></i>	<i>A1</i>	<i>A2</i>	<i>A3</i>
000001	00000000001	00000000010	00000000011

means: "Add (code in AOp) the number stored in the first location of the memory unit (code in A1) to the number stored in the second location (code in A2), and transfer their sum to the third location (code in A3)".

The definite harmonizing of the decoder of the operations commutator, in one of whose output buses the control voltage is produced, corresponds to the recording of the instruction code in the ISU. This voltage prepares the loops in the operations-control unit which are required for performance of the given operation.

The first stage of performance of the given operation is the readout of the code of the first number from the working memory unit and its write-in into one of the registers of the arithmetic unit. The readout takes place due to the fact that the pulse from the central control unit, passing through the prepared loops of the COC unit, registers the number code of the first number from A1 with its transfer to the working storage of the control unit. The writing of the code of the first number into the corresponding register of the arithmetic unit takes place as a result of the fact that the input gates of the register connecting the code buses to its flip-flops are triggered by a pulse from the central control unit at the moment of readout of the code of the first number from the memory unit.

After the first number has been received in the register of the arithmetic unit from the central control unit CCU, the next control pulse is issued, causing the second number from the working storage to be recorded in the second register of the arithmetic unit. /388

The next stage is the immediate performance of the given operation. If the operation is carried out with central control, then all necessary control pulses and potentials are fed to the control element of the arithmetic unit across the prepared loops of the COC unit from the CCU. If the operation is carried out with local control, then the COC and the CCU are switched off and all necessary control pulses and potentials are fed to the control element of the arithmetic unit from the LOC unit. After performance of the operation, the result of the calculations is fed to the arithmetic unit.

The rewrite of the result from the arithmetic unit to the memory unit, in accordance with the address indicated in A3 of the instruction storage unit, takes place in the following manner: The corresponding pulse from the CCU which, in the case of control of the operations with the LOC unit, is switched on immediately after the end of the operation, copies the number code with A3 from the instruction-storage unit by means of the loops of the COC unit and transfers it along the address code buses ACB to the control element of the working memory unit. This makes it possible to select the corresponding cell of the working storage in which the result of the effected operation must be re-

corded. At the same time, the code of the number result is fed to the working memory unit from the arithmetic unit and is stored in the selected cell.

This completes the execution of the given instruction, and the computer proceeds to execution of the next instruction whose code must be first placed in the instruction-storage unit ISU. When the instructions are followed in their number sequence (which is true in most cases), their codes are rather easily selected from the working storage with the aid of one of the instruction-control units (CIC or LIC).

If the CIC unit controls the readout of the instructions from the memory, then, after execution of the given instruction, a pulse is fed to the input of this unit which is a flip-flop counter. This corresponds to increasing the stored code by unity. Thus, the number code of the next instruction in sequence is formed in the CIC unit. In accordance with this code, the code of the next instruction is selected from the working storage and is placed in the ISU.

If the LIC unit controls the readout, then everything proceeds as in the case of control by the CIC unit, except that the pulse of the increase of the instruction number code by unity is fed to the input of the LIC unit.

Occasionally, the sequence of instructions is disrupted. This happens when the subsequent computational program depends on the results of the first /389 stage: If this result is greater than a certain constant number, the subsequent calculations must proceed in accordance with one program; if it is less - in accordance with another. The possibility of branching of the calculation process is considered with the aid of a comparison instruction from which the correct direction of the subsequent calculations is automatically selected.

The operation code in the comparison instruction which can be written in the AOp of the instruction storage unit indicates that the comparison operation should be performed and, depending on the result of such comparison, one or the other instruction sequence should be selected. By means of the operations commutator, this prepares the loops not only for performance of this operation but also for registration of its results.

The number codes of the working memory locations where the numbers to be compared are stored, are written into A1 and A2 of the instruction storage. The number code of the working memory location where the instruction code for initiating the computation is stored, is written into A3 if the first of the numbers to be compared is less than the second.

The comparison operation is effected upon instructions from the CCU. The instruction pulses from the CCU initiate the sequential transfer of the number codes from the memory locations where the numbers are stored to the control unit of the working storage, and cause the readout of these numbers and their write-in into the registers of the arithmetic unit. The arithmetic unit compares the numbers and, in accordance with the result of the comparison, excites the corresponding circuits in the COC by means of its control element.

If the first number is less than the second, those circuits are excited which erase the code contained in the CIC unit and transfer the code from the A3

instruction-storage unit to the CIC. The subsequent calculation process begins with execution of the instruction whose number code was recorded in A3.

If the first number is greater than the second or equal to it, then other circuits are excited, the code in the CIC unit is not erased, and unity is added to the code contained in the CIC unit. In fact, the next instruction in sequence of the basic program is executed.

The operation of the control unit during execution of the instructions in natural sequence and during execution of the comparison instructions is most characteristic. In executing instructions in cases in which, for example, the number groups are transferred from the memory unit to the working storage on magnetic tape or during transfer from the basic program to the subprogram, the elements of the control unit operate in similar fashion.

### Section 57. The Control Units of One-Address Computers

The control units of one-address computers are generally simpler in design than the corresponding units of three-address computers since, in this case, 390 fewer elementary functions are required to execute one instruction. A one-address instruction consists only of two parts: the operation code and the address indicating the memory location which stores one of the numbers taking part in the operation. The second number participating in the operation usually is already contained in the adder of the arithmetic unit before execution of the instruction begins.

A one-address instruction written in the octal system of notation as 01 0102 may mean: Add the number stored in the memory location numbered 0102 to the number stored in the adder; the result of execution of the operation is stored in the adder.

During performance of a one-address arithmetic operation, only a single exchange of information occurs between the memory unit and the arithmetic unit while such an exchange takes place three times in three-address computers. This permits some simplification of the control units of one-address computers. In addition, the one-address instruction itself has not more than half the number of digits of a three-address computer. This permits a considerable reduction in necessary hardware for construction of the main register of the control unit or for the instruction-storage unit.

Depending on the control principle (hybrid or centralized), the control units in one-address computers may have different circuits. With hybrid control, the unit usually consists of several functional blocks that are similar in function to the elements of the control unit in a three-address computer. With centralized control, the basic elements of the unit ordinarily consist of diode or magnetic matrices which produce almost all the control pulses necessary to execute the instructions.

Figure 227 shows a variant of the block diagram of the control unit in a one-address computer with hybrid control. The unit consists of the following building blocks: instruction register IR, instruction decoder ID, control

counter CC, local operation control LOC, pulse generator PG, main pulse generator MPG, and control console CP.

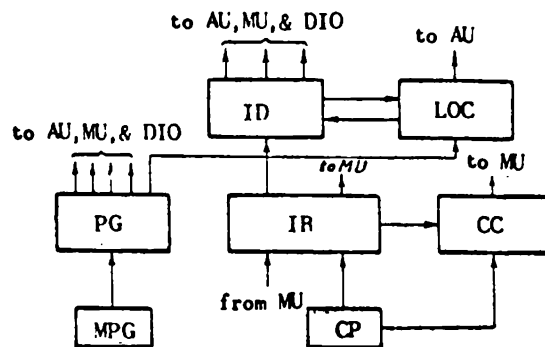


Fig.227 Control Unit of a One-Address Computer with Hybrid Control

The instruction register IR receives the instruction code from the memory unit and stores it; this instruction will be executed during the next operation cycle of the computer. At times, this register is also used for changing the instruction codes during cyclic repetitions of the individual parts of the problem-solving program (e.g., in the "Ural" computers). The basic element of this register is the main instruction register divided into two parts, one <sup>/391</sup> for storage of the operation code and the other for storage of the code of the address part of the instruction. If a possibility is provided for changing the instructions in the unit under consideration, then the main instruction register performs in accordance with the circuit of the adder.

The instruction register is connected with the instruction decoder to which the operation code is transmitted and with the control counter to which either a single pulse or the address part of the modified instruction is fed in each cycle. The register is also connected to the pulse generator which generates a train of control pulses and with the control console. The connection with the console provides for writing of the first instruction of the program into the main register and for control of the operation of the units under various conditions.

The instruction decoder ID converts the operation code into a control signal which is obtained at the corresponding output bus. The basic elements comprise the register to which the operation code is transferred from the instruction register and the decoder (diode matrix or electromagnetic).

The output buses of this building block are the output buses of the decoder which are connected, over decoupling and amplifying elements, to the corresponding input buses of the local controls of the arithmetic and memory units as well as to the data input and output units. Moreover, some of the outputs of the decoder are connected to the local operation control.

The control counter CC gives the sequence of execution of the instructions



and, if necessary, changes this sequence during solution of the problem. The basic element of this functional block is the counter itself which has a number of columns corresponding to the number of memory locations that store the instructions. Thus, if the memory unit has 2048 locations, the counter should be of the 11-place type.

In the natural sequence of execution of the instructions, the contents of the counter is increased in each cycle by unity. Since the number fixed by the counter serves as the address of the next instruction, then during its transfer to the local control of the memory unit, the necessary instruction will be selected from the latter and transferred to the instruction register. When it is necessary to change the natural instruction sequence, a number is written into the control counter, which is initially set to zero, from the address part of the main instruction register.

The local operation control LOC produces several control pulses during performance of the arithmetic operations which are not included in the basic cycle of the computer. Thus, the operation of division can be transferred to local/392 control; in some cases, the time of performance of this operation is ten times greater than the time required for performing the operation of multiplication.

Operation of the LOC is controlled by the signals arriving from the instruction decoder; the reference pulses are fed from the pulse generator. From the outputs of the local operation control, the control pulses enter the local control of the arithmetic unit.

The pulse generator PG produces a train of pulses fed in a definite sequence to the inputs of the local controls of the arithmetic and memory units, as well as to the data input and output units. The pulses of these trains, in accordance with the signals obtained from the output of the instruction decoder, are used to generate control pulses which are fed directly to the groups of elements (gates, flip-flops, etc.) of the computer units.

The pulse generator is a pulse distributor consisting of gates, delay lines, and other elements. Its reference pulses are supplied by the main pulse generator which is usually of the quartz oscillator type.

The console CP is designed for manual control of the computer. In addition to the elements used to start and stop the computer and to change its operating conditions, the console is equipped with a signaling system designed for visual control of the operation of the computer units.

In the natural sequence of execution of the instructions, the unit in question operates in the following manner: In each cycle, the contents of the control counter is increased by unity. This leads to readout of the instructions from the memory locations which are numbered consecutively. During the cycle, the respective instruction is selected from the memory unit and executed, after which the result is usually stored in the adder of the arithmetic unit.

The instruction selected from the memory unit, in accordance with the address corresponding to the number in the control counter, is transferred within

the control unit to the instruction register. From there, the operation code is fed to the instruction decoder, and the address part of the instruction is fed to the number-address element of the memory unit. The required number is extracted from the memory unit, and operations determined by the operation code are performed on it.

The operation code is decoded in the instruction decoder. The signal produced at one of the decoder outputs prepares the corresponding networks of the local controls for reception of the pulses from the pulse generator. Thus, during addition, pulses from the pulse generator enter only those circuits of the local controls of the arithmetic and memory units which generate the control pulses for the readout of one number from the memory unit, its transfer to the arithmetic unit, and its addition to the number already stored in the adder.

During execution of the instruction, the contents of the control counter 393 is increased by one. The processing of the program can be accelerated by determining the operating condition of the unit at which the next instruction in sequence is selected from the memory unit and fed to the instruction register even before execution of the previous instruction is completed.

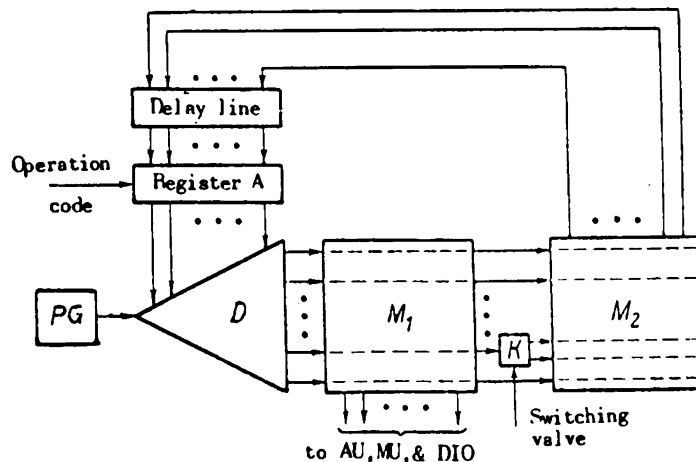


Fig.228 Control Unit with Diode Matrices

With centralized control, the control unit of a one-address computer, as noted above, can be built on the basis of diode or magnetic matrices. Diode matrices are used together with vacuum-tube circuits since they ensure reliable operation only when the effective potentials have large values. Magnetic matrices can be used for almost all circuits of an electron digital computer, including ferrite-transistor types.

The basic elements of a control unit with diode matrices are shown in Fig.228. The unit consists of the following: operation code decoder  $D$ , control diode matrix  $M_1$ , serial matrix (diode)  $M_2$ , pulse generator  $PG$ , and a switching valve for the input loops of the serial matrix  $K$ .

The control matrix generates the control signals required for performing

the operations on the numbers. These signals are fed directly to the controlled elements and element groups of the arithmetic and memory units as well as to the data input - output units. The signals are generated by excitation of certain matrix buses connected to the outputs of the decoder.

The serial matrix is used for a sequential generation of control pulses during the performance of some operation. Use of this matrix is necessary in view of the fact that, during performance of any operation, several series <sup>/394</sup> of control pulses are required at various instants of time while the matrix  $M_1$ , when the contents of the register A is constant, can produce only one series of pulses.

The serial matrix and the control matrix consist of a series of horizontal and vertical buses interconnected by diodes, similar to the arrangement in diode decoders. The horizontal buses of the matrices are connected to the decoder outputs and trains of control pulses are produced in the vertical buses.

The pulse generator determines the frequency of generation of the control pulse train. From here, the pulses are fed to the decoder and, in accordance with the contents of the register A, appear at one or another decoder output, i.e., excite one or the other horizontal matrix buses.

The unit operates as follows: The operation code, as one of the component parts of the next instruction, is fed to the register A and decoded by the decoder D so that the pulse from the pulse generator PG passes to its prescribed output. This causes excitation of one of the horizontal matrix buses, and the first series of control pulses is produced at the outputs of the control matrix which are connected to the excited bus across the diodes.

To generate the next series of control pulses used in execution of a given instruction, it is necessary to change the contents of the register A, i.e., to insert several conditional codes. These codes, like the pulse trains, appear at the outputs of the serial matrix. In order that the contents of the register A change only after completion of all the operations connected with the generation of the previous control pulse train, the codes from the output of the matrix  $M_2$  are fed through the delay line to the register.

Thus, by the time the next pulse from the pulse generator arrives at the decoder input, the new value of the code is fixed in the register A, which finally leads to excitation of a horizontal bus of the matrices other than that in the previous cycle. The next train of control pulses appears at the outputs of the matrix  $M_1$ , while the next code, which is fed to the register A, is produced at the outputs of the matrix  $M_2$ .

On generation of the last series of control pulses necessary for execution of a given instruction, no code appears at the outputs of the matrix  $M_2$  since none of these outputs is connected to the excited bus, and the operation code of the subsequent instruction is fed to the register A.

The sequence of generation of the control pulses may change even during execution of one instruction. Thus, in multiplication, the generation of control signals depends on the values of the multiplier digits that show the magnitude

of the corresponding partial product. To change the sequence of generation of the control pulses, one or several valves K are used for switching the input <sup>/395</sup> circuits of the matrix  $M_2$  (for example, relative to the values of the multiplier digits).

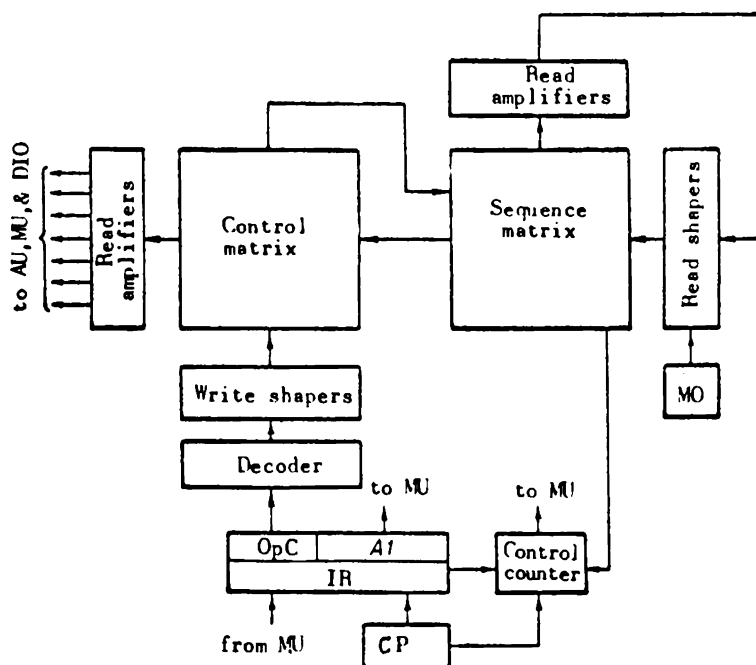


Fig.229 Control Unit with Magnetic Matrices

In using magnetic matrices, the basic principle of generation of the control pulses is the same as in the case of diode matrices. However, because of the peculiarities of magnetic matrices, the circuit of the unit is somewhat simplified due to the insertion of a direct connection between both matrices.

The block diagram of a control unit with magnetic matrices is shown in Fig.229. The unit consists of the following: instruction register IR divided into two parts (for spacing the operation code OpC and the address part of the instruction AI), a master oscillator MO and a console CP.

All control pulses are produced at the outputs of the read amplifiers which are connected to the control matrix. The sequence of generation of these pulses is determined by a signal from one of the decoder outputs; this signal corresponds to the operation being performed. The direct production of the necessary series of control pulses takes place under the action of the signals produced in the serial matrix.

Both matrices are a system of ferrite number lines connected to the de- <sup>/396</sup> coder outputs by the write shapers. The number of such lines in the matrix is usually equal to the number of the various arithmetic and logical operations

performed in the computer. In other words, each number line of the control matrix is used for the generation of control pulses during performance of a given operation. In some cases, it is possible to use one number line for the generation of pulses controlling the performance of several operations.

The number lines of the serial matrix are connected to the read shapers by means of the read amplifiers, due to which the necessary sequence of generation of the control pulses is established.

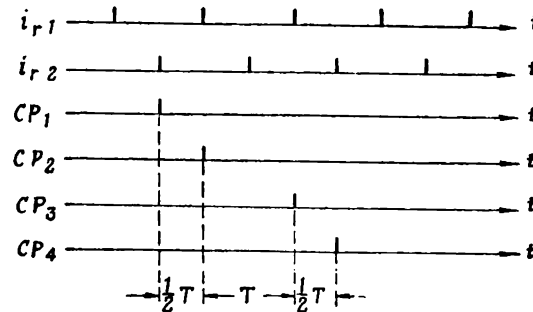


Fig.230 Timing Chart for Control Pulse Generation

The master oscillator MO usually produces clock pulses of two series:  $i_{r1}$  and  $i_{r2}$ . These pulses are continuously fed to the read shapers, thus participating in the assignment of a definite sequence for the generation of the control pulses.

In the next example, we will examine the operation of magnetic matrices during execution of an instruction inserted in the instruction register unit. Let an instruction with the operation code 04 be inserted in the instruction register IR. To execute this instruction, it is necessary to generate four control pulses  $CP_1 - CP_4$  distributed in time as shown in Fig.230. For this purpose, there is one number line each in the composite of every matrix. All lines are connected in series (Fig.231). The number line of the control matrix consists of four ferrite cores (shown as rectangles) and the number line of the serial matrix consists of five cores. The write bus, connected to the output of the shaping cell (the shaper)  $SC_{04}$ , passes through all ferrite cores.

The output windings of the ferrite cores of the number lines are shown by vertical lines and the read windings, by horizontal lines. The output windings are connected to the inputs of the corresponding amplifiers A, and the read windings are connected to the outputs of the read shapers  $SC_1 - SC_5$ . Clock pulses of the series  $i_{r1}$  and  $i_{r2}$  are fed to the read inputs of these shapers, while the write inputs are fed with pulses from the outputs of the read amplifiers  $A'_1 - A'_5$  of the serial matrix.

When the operation code is decoded at the decoder output to which the shaper  $SC_{04}$  is connected, a pulse is produced. The shaper  $SC_{04}$  amplifies this pulse and switches all the ferrite cores of the number lines to the state 1. /397

After this, the write pulse WP is fed to the input of the shaper  $SC_1$ ; this pulse is produced in the serial matrix within a definite interval of time after completion of the previous operation.

The next clock pulse of the series  $i_{r1}$  resets the ferrite core of the shaper  $SC_1$  to the state 0, as a result of which the ferrite core  $FC'_1$  of the number line of the serial matrix also is switched to the zero state. The pulse produced in the output winding of the ferrite core  $FC'_1$ , is amplified by the amplifier  $A'_1$  and then fed to the input of the shaper  $SC_2$ . The next clock pulse of the series  $i_{r2}$  resets the shaper to the state 0.

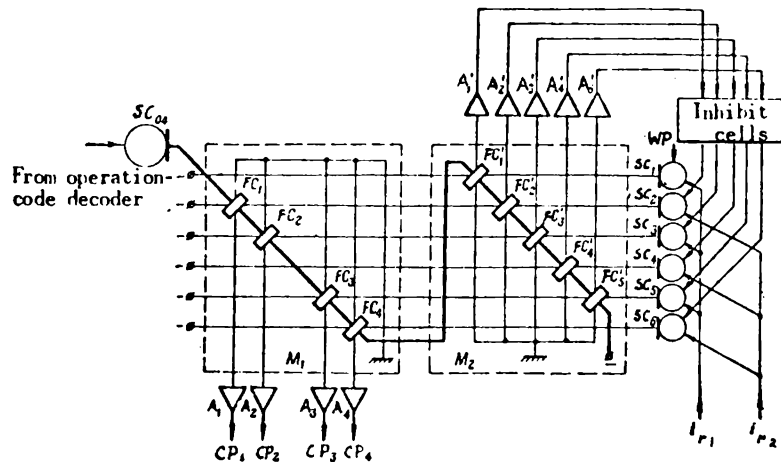


Fig.231 Circuit of Part of the Control Unit Components

During switching of the shaper  $SC_2$  to the zero state, a read pulse is produced at its output; in turn, this pulse switches the ferrite core  $FC_1$  of the control matrix and the ferrite core  $FC'_2$  of the serial matrix to the state 0. The pulses generated in the output windings of these ferrite cores are amplified, producing the control pulse  $CP_1$  at the output of the amplifier  $A_1$ , and the pulse ensuring further operation of the unit, at the output of the amplifier  $A'_2$ .

Generation of the other control pulses proceeds in a similar manner and in complete accord with the timing chart in Fig.230. After generation of the last control pulse  $CP_4$  in the given series, the operation code of the next instruction is fed to the inputs of the decoder, which finally causes the excitation/398 of the corresponding write shaper and of the number lines of the matrices connected to it. Then the next instruction is executed, and so on until all the instructions in the program have been executed.

## Section 58. "Hard" Program Systems

"Hard" program systems are designed for single or repeated generation of a definite sequence of control signals. They can be used like the central control unit of special-purpose computers and like the local controls of various types of computers. "Hard" program systems are made either in the form of com-

plex delay lines with numerous outputs (according to the number of control signals being generated), or in the form of functional blocks with magnetic matrices. The switching of the elements comprising a "hard" program system is generally constant. This switching is effected in accordance with a predetermined timing chart for generation of the control signals necessary for problem-solving or for performing some operation.

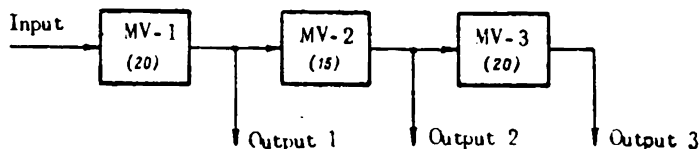


Fig.232 Delay Lines in Three One-Shot Multivibrators

A delay line with several outputs may consist of one-shot multivibrators which have the same or a different latency time for the pulses arriving at their inputs. Figure 232 gives the circuit diagram of a delay line made of three one-shot multivibrators. The first and third monostable multivibrators (MV-1, MV-3) delay the pulse for 20  $\mu$ sec and the second vibrator (MV-2), for 15  $\mu$ sec. At the first output, the pulse emerges within 20  $\mu$ sec after its arrival at the input of the line; at the second output, within 35  $\mu$ sec; and at the third output, within 55  $\mu$ sec. If a certain group of gates must be triggered every 35  $\mu$ sec, they can be controlled by pulses leaving the first and third outputs of the delay line. In simultaneous control of several elements, the pulses produced at the outputs of the monostable multivibrators are amplified by special amplifiers or blocking oscillators.

In the following example, we will consider the construction of a more complex "hard" program system on monostable multivibrators. Let it be required, at a frequency of  $f = 6.67$  kc, to construct the sequence of the control pulses distributed along the control loops in the following manner:

/399

- a pulse in the control loop 1;
- within 5  $\mu$ sec - in loop 2;
- within another 20  $\mu$ sec - in loop 2;
- within another 20  $\mu$ sec - in loop 2;
- within another 20  $\mu$ sec - in loops 4 and 3;
- within another 20  $\mu$ sec - in loop 4;
- within another 20  $\mu$ sec - in loop 4;
- within another 20  $\mu$ sec - in loop 4;
- within another 20  $\mu$ sec - in loops 6 and 5;
- within another 5  $\mu$ sec - in loop 7.

This sequence of generation of the control pulses is effected with the aid of a unit which comprises nine monostable multivibrators. The circuit diagram of this unit is given in Fig.233. For seven of the vibrators, the latency time of the pulses arriving at their inputs is equal to 20  $\mu$ sec, while for the other two the latency time is 5  $\mu$ sec. Pulses with a repetition rate of 6.67 kc are

fed to the input of the first multivibrator. These pulses are directly fed to the first control loop. The other control loops are fed with pulses from the outputs of the corresponding multivibrators. It should be noted that some or all of the pulses fed to the control loops are first amplified and shaped.

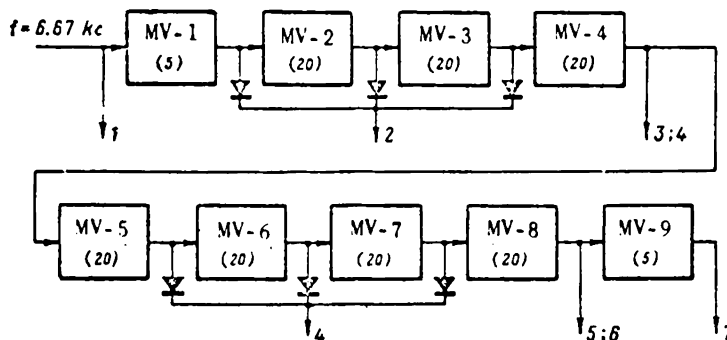


Fig.233 "Hard" Program System in One-Shot Multivibrators

The control delay line may also consist of flip-flops. In this case, it comprises a shift register whose flip-flops are fed shift pulses of different frequencies.

Figure 234 shows the circuit diagram of an elementary control delay line based on three flip-flops. The flip-flops are connected in series. Their pulse outputs, together with the loop connected to the input of the flip-flop  $T_1$ , form the four control outputs of the entire line. Pulses of different frequencies/400 are fed to the input of the flip-flop  $T_1$  and to the reset loops of the flip-flop to make sure that the pulses received at these outputs have the required time-sequential distribution. Moreover, the rate of the pulses fed to the input of the flip-flop  $T_1$  should be less than the rate of the pulses fed to the flip-flop reset loops.

Let us consider the case when the input of the flip-flop  $T_1$  is fed with pulses having a repetition rate of  $f_3 = 250$  kc; the reset loop of the flip-flops  $T_1$  and  $T_2$ , with pulses of a repetition rate of  $f_2 = 1$  kc; and the reset loop of the flip-flop  $T_3$ , with pulses of a repetition rate of  $f_1 = 2$  kc. The pulses arrive from a certain flip-flop divider.

The pulse with a rate of  $f_3$  arriving at the input of the flip-flop  $T_1$  switches this flip-flop to the state 1 and simultaneously arrives at the first output of the line. After  $1 \mu\text{sec}$ , the pulse with a rate of  $f_2$ , which entered the reset loop of the flip-flops  $T_1$  and  $T_2$ , resets the flip-flop  $T_1$  to the state 0, which causes the production of a pulse at the second output of the line and at the input of the flip-flop  $T_2$ . The reset pulse which had switched the flip-flop  $T_1$  to the state 0, also enters the flip-flop  $T_2$ . However, this pulse does not inhibit the setting of the flip-flop  $T_2$  to the state 1 by the transfer pulse from the flip-flop  $T_1$  since, being delayed somewhat in the transfer loop, this latter pulse enters flip-flop  $T_2$  somewhat later than the reset pulse. After another  $1 \mu\text{sec}$ , the flip-flop  $T_2$  is reset to the zero state by the next



pulse with a rate of  $f_2$ . This causes the arrival of a pulse at the third output of the line and the switching of the flip-flop  $T_3$  to the state 1. After another  $0.5 \mu\text{sec}$ , the pulse with a rate of  $f_1$  resets the flip-flop  $T_3$  to the zero state, causing the production of a pulse at the fourth output of the line. The process of formation of the control pulses is repeated as shown in Fig.235.

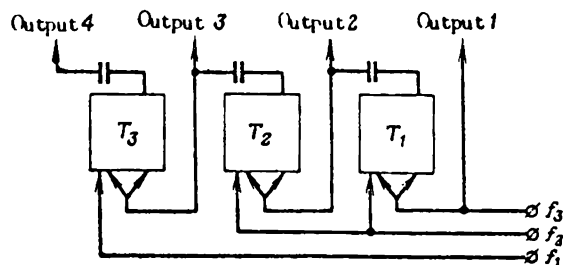


Fig.234 Delay Line Based on Flip-Flops

Let us construct a flip-flop delay line for the case of generation of control pulses, which was studied above in the construction of the control delay line based on monostable multivibrators. The control should be periodic with a frequency of  $f = 6.67 \text{ kc}$  at a transfer of control pulses after 5 and  $20 \mu\text{sec}$ . This means that pulses with a repetition rate of  $f_3 = 6.67 \text{ kc}$  should be fed to the input of the first flip-flop of the control delay line while the flip-flop reset loops should be fed with pulses of a rate of  $f_1 = 200 \text{ kc}$  and  $f_2 = 50 \text{ kc}$ .

It is easy to obtain the frequency  $f_2$  by a successive division of the frequency  $f_1$  by two. The frequency  $f_3$  cannot be obtained in this manner; thus, to simplify the control circuit (its divider), a frequency  $f'_3$  which is close to 401 the frequency  $f_3$  must be used instead of  $f_3$ ; this can be readily effected by division of the frequency  $f_1$ . Moreover, the frequency  $f'_3$  should be lower than the frequency  $f_3$  since  $f_3$  is the maximum operating frequency of the unit under study.

On the basis of the above statements, it is readily established that the frequency  $f'_3 = 6.25 \text{ kc}$ ; this is obtained by dividing the frequency  $f_1$  five times by two. Thus, to feed the control delay line, a pulse generator with a frequency of  $f_1 = 200 \text{ kc}$  and a divider consisting of five flip-flops are required in this case. The delay line itself consists of nine flip-flops connected in series. Pulses with a repetition rate of  $f'_3 = 6.25 \text{ kc}$  are fed to the input of the first flip-flop; the reset loops of the first and ninth flip-flops are fed with pulses having a repetition rate of  $f_1 = 200 \text{ kc}$ ; the reset loops of the other flip-flops are fed with pulses at a rate of  $f_2 = 50 \text{ kc}$ .

Figure 236 shows the general circuit of the given control delay line based on flip-flops, together with the frequency divider. The frequency divider consisting of the flip-flops  $T_{10} - T_{14}$  is supplied from the pulse generator PG. The flip-flops  $T_1 - T_9$  form the delay line itself. In the diagram, the numerals in parentheses next to the labeling of the outputs of the delay line represent the control unit loops to which the corresponding delay lines are connected.

It is possible to construct "hard" program systems in the form of delay lines not only for the flip-flop circuits but also for circuits based on ferrite-transistor cells. In this case, the control delay lines consist of ferrite-transistor cells. This makes it possible to obtain the necessary matching of the control and controlled circuits.

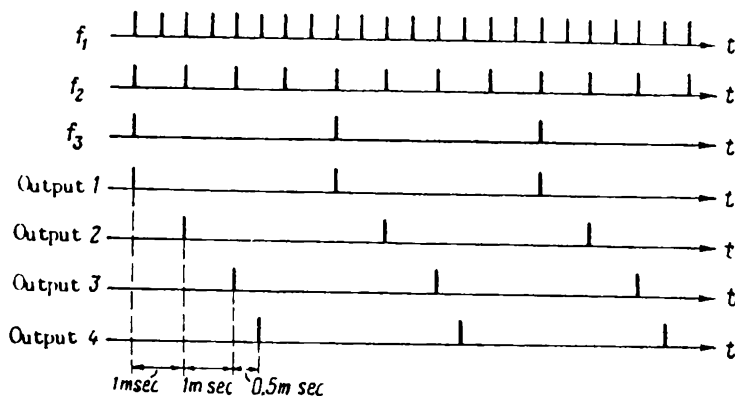


Fig.235 Timing Chart for Operation of the Flip-Flop Delay Line

It is known that two ferrite-transistor cells, connected in series which are fed with clock pulses of one frequency but shifted by  $180^\circ$  in phase with 402 respect to the other, will delay the pulse for a time corresponding to the period of the clock pulses, or for one cycle. The use of long delay lines made of ferrite-transistor cells connected in series makes it possible to produce control pulses at the required instants of time similar to the manner used for the delay lines in one-shot multivibrators and flip-flops.

Figure 237 shows the circuit of a delay line used for producing control pulses in the zero (at the initial instant), second, and fifth cycles. This same Figure also gives the timing chart for operation of the line.

If the solution of the problem takes comparatively more time, it is not recommended to use a control delay line fed by clock pulses of one frequency. This is based on the fact that, in these cases, the control delay line should contain a large number of elements. For example, if the problem is solved in 2010 cycles and the last control pulse must be fed in the 2000th cycle, then a control line consisting of 4000 ferrite-transistor cells would be necessary.

A reduction in the number of ferrite-transistor cells required for construction of a control delay line is possible with the use of clock pulses 403 of different frequencies. Moreover, the "hard" program system consists of a delay line and a frequency divider. The frequency divider is constructed of elementary binary dividers (frequency dividers in twos). Each of these includes three ferrite-transistor cells, one of which is the inhibit cell.

The circuit of the elementary frequency divider is given in Fig.238 and the timing chart for its operation, in Fig.239. The input of the cell 1 is fed with pulses of the series  $i_{r,1}$ ; the read winding of this cell and the cell 3 are fed

with pulses of the series  $i_{r2}$ ; and the read winding of the cell 2 is fed with pulses of the series  $i_{r1}$ . Because of this distribution of the pulses of the series  $i_{r1}$  and  $i_{r2}$ , pulse trains whose frequency is halved are produced at the

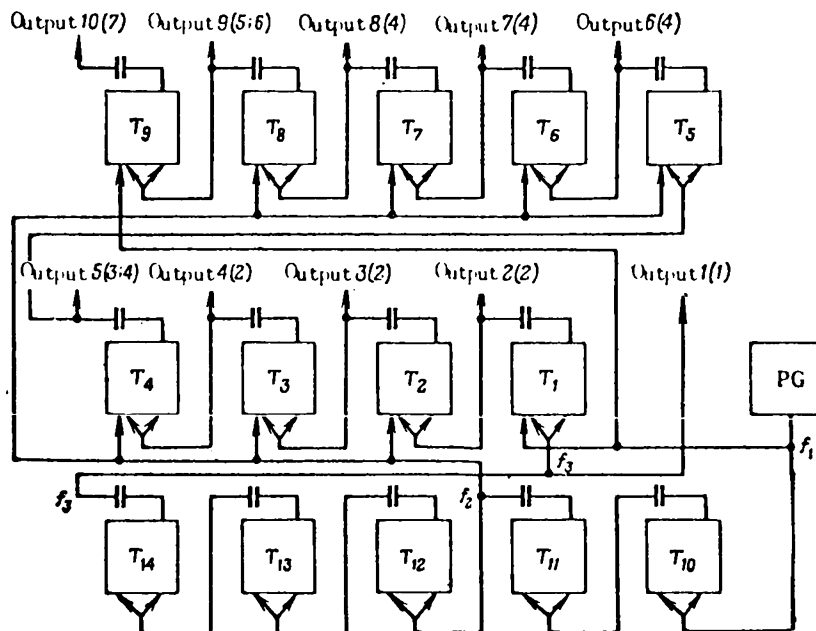


Fig.236 "Hard" Program System Based on Flip-Flops

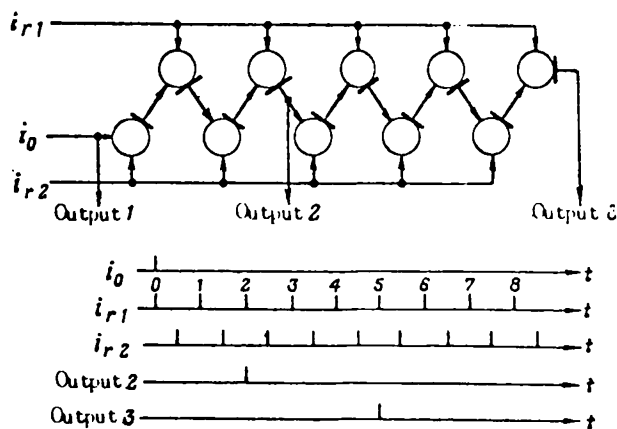


Fig.237 Delay Line Based on Ferrite-Transistor Cells and Timing Chart of its Operation

outputs of the cells 2 and 3; these pulses are synchronized with the pulses /404 of the series  $i_{r1}$  and  $i_{r2}$  respectively.

By connecting several binary dividers in series, pulses can be obtained whose repetition rate is 2, 4, 8, etc. times lower than the repetition rate of

the basic read pulses of the series  $i_{r1}$  and  $i_{r2}$ . The timing chart of the pulses produced at the output of the divider which divides the frequency of the basic read pulses by 2, 4, 8, 16, and 32, is given in Fig.240. The frequency of the basic read pulses is denoted in the timing chart by  $f$  ( $f_1$  or  $f_2$ ).

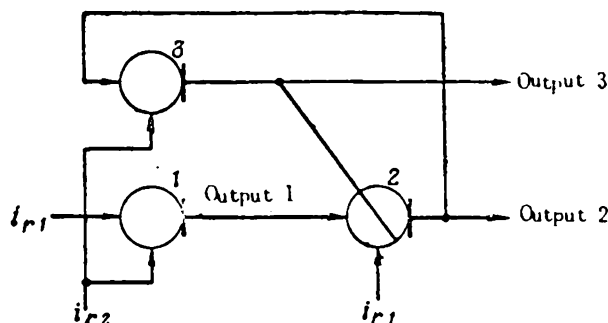


Fig.238 Binary Divider Based on Ferrite-Transistor Cells

Let us assume that, for normal operation of a certain digital computer, it is necessary to feed its units with control pulses at the 1st, 8th, 12th, 16th, 30th, 32nd, 64th, 72nd, and 96th cycles. Then, a pulse must be fed to the input of the delay line in the 1st cycle. The delay line itself is constructed according to the principle of greatest reduction in the total number of ferrite-

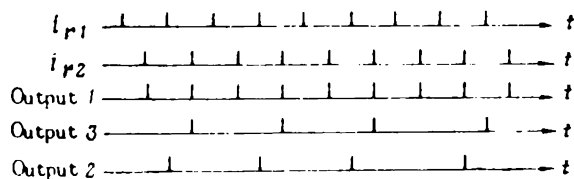


Fig.239 Timing Chart for Operation of a Binary Divider

transistor cells.

Figure 241 shows the circuit of a "hard" program system which generates control pulses at the instants of time required for the given case. The system contains a master oscillator MO generating the basic clock pulses with a fundamental frequency of  $f$ , a frequency divider, and the delay line itself consisting of 17 ferrite-transistor cells. The control pulses are taken from the outputs of the 2nd, 4th, 5th, 11th, 12th, 14th, 16th, and 17th cells; the control pulse of the first clock is the input pulse  $I_1(t_1)$  for the delay line.

A study of the timing chart of the clock pulses (Fig.240) readily shows that a pulse will be generated in the 8th cycle at the output of the 2nd cell; in the 12th cycle, at the output of the 4th cell; in the 16th cycle, at the output

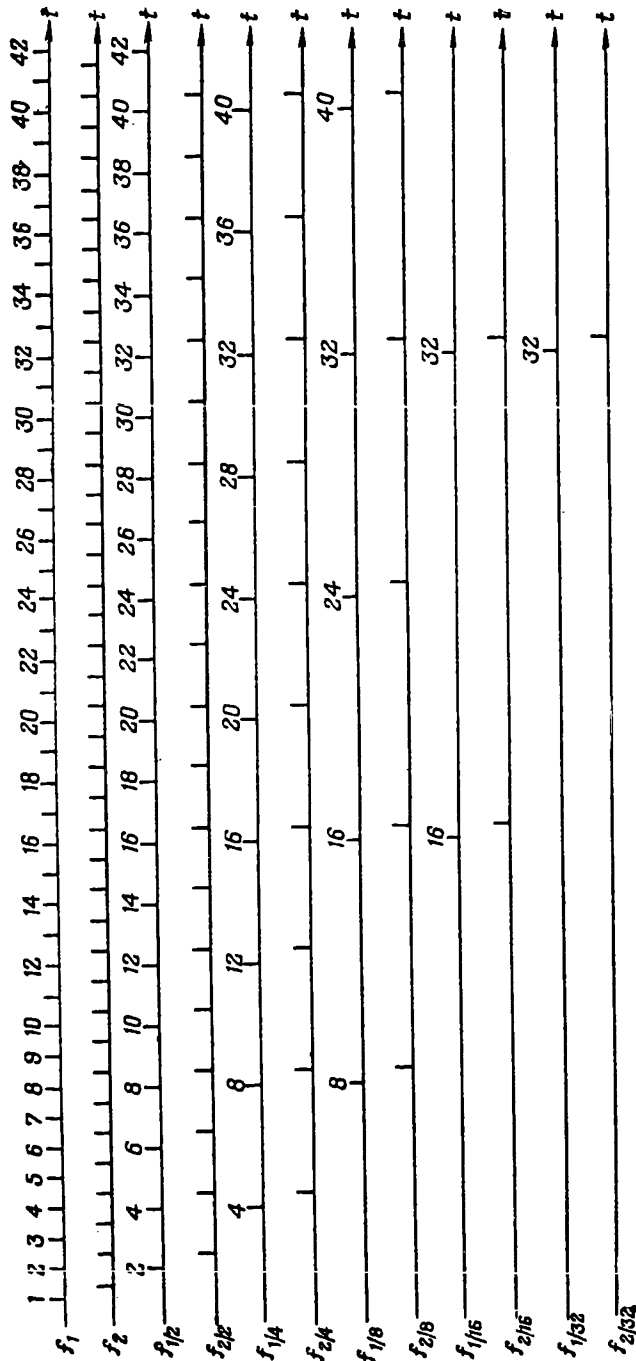


Fig.240 Timing Chart for Operation of a 2, 4, 16, and 32 Divider

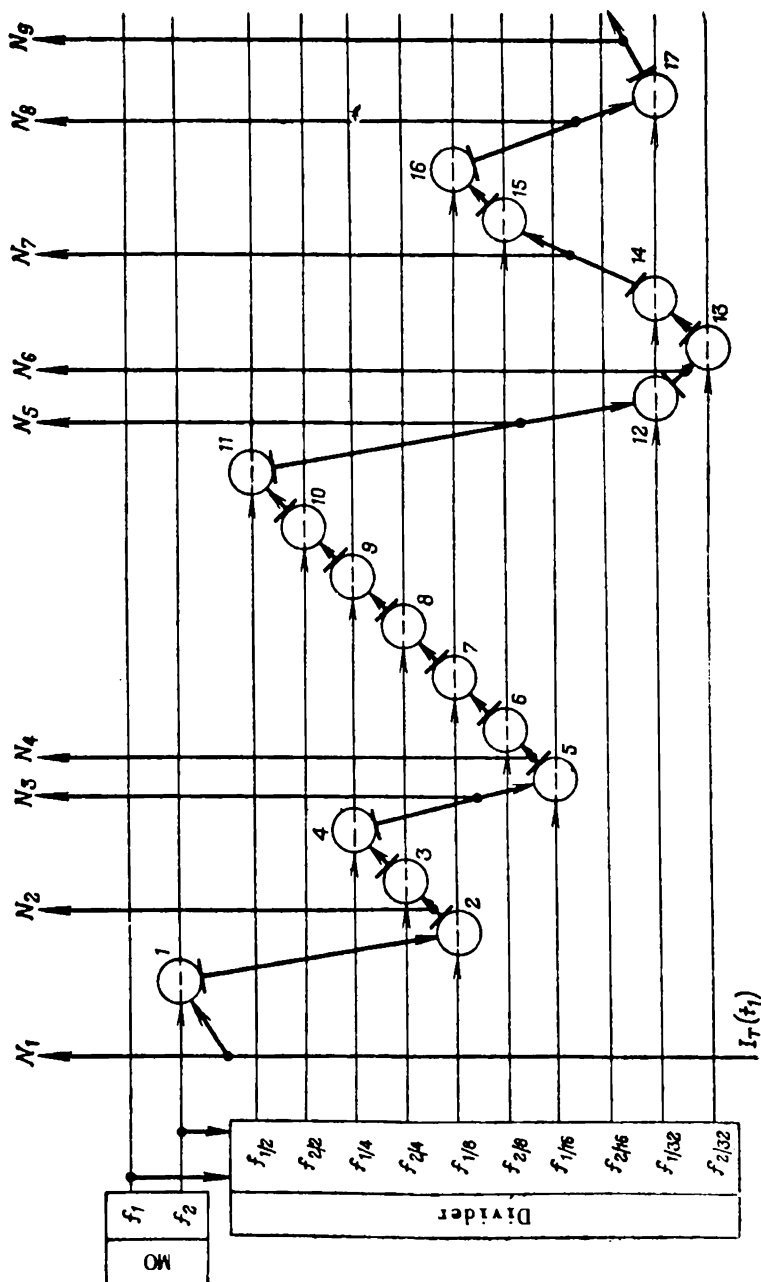


Fig. 241 "Hard" Program System in Ferrite-Transistor Cells

of the 5th cell, etc. The greatest reduction in the number of cells in the control delay line is achieved when the next control pulse is generated at the output of the cell fed by clock pulses of a lower frequency. Let us study this last statement in more detail.

The fourth control pulse (instruction)  $N_4$  is produced at the output of the 5th cell in the 16th cycle. The next instruction  $N_5$  should be produced in the 30th cycle. In this cycle, the divider issues one of the clock pulses of frequency  $f_{1/2}$ , which should also be used to produce the instruction  $N_5$  at the output of some ferrite-transistor cell. Since the code "1" can be written into this cell only immediately before admission of the clock pulse in the 30th 407 cycle, a delay line of five cells must be inserted between it and the cell giving the instruction  $N_4$ .

In transferring from instruction  $N_3$  to instruction  $N_4$ , as well as from instruction  $N_5$  to instruction  $N_6$  and from  $N_6$  to  $N_7$ , the intermediate cells are not used. Thus, if in the transfer from instruction to instruction, the frequency of the clock pulses decreases, the number of intermediate cells will be minimal, and vice versa.

This example of the construction of a "hard" program system graphically shows the possibilities of reducing the bulk of the unit when clock pulses of different frequencies are used. Actually, 192 ferrite-transistor cells would be necessary for an ordinary delay line, since the last instruction is issued in the 96th cycle. When clock pulses of different frequencies are used, the number of cells in the delay line itself is reduced to 17. In that case, 32 cells are required in all, since the frequency divider consists of 15 ferrite-transistor cells.

If it is necessary to issue a large number of instructions (hundreds and thousands) for the solution of a given problem, then the "hard" program system, constructed of a delay line with the use of clock pulses of different frequencies, is very cumbersome and requires a large number of semiconductor triodes. To reduce the bulk of the unit, especially to reduce the number of transistors, ferrite number lines consisting of a magnetic matrix can be used in the "hard" program systems.

In addition to the matrix, the system contains a master oscillator and a frequency divider similar to the building blocks used for the control delay line, two ferrite-transistor shift registers, and amplifiers.

The shift registers are ordinary delay lines based on ferrite-transistor cells with outputs after each cell. The first shift register is fed with clock pulses whose frequency is several times lower than the frequency of the clock pulses feeding the second shift register. The frequency ratio is so selected that, in an interval of time limited by the instants of exit of the pulses at two adjacent outputs of the first shift register, pulses are produced at each of the outputs of the second shift register. The initial write-in of the code "1" into the shift registers takes place when pulses of corresponding frequencies are emitted by the divider.

The matrix consists of ferrite number lines, similar to the lines used in

the permanent storage units. The codes in the lines are recorded in pulses from the outputs of the second shift register. The codes read out of the matrix are original instruction codes.

In the following example, let us consider the operation of a "hard" program system with a magnetic matrix. Let it be necessary to issue 10 instructions during each 16 cycles of operation of the computer in the following sequence: /408

- in the 1st cycle - instruction  $N_1$  with the code 1011;
- in the 2nd cycle - instruction  $N_2$  with the code 1001;
- in the 4th cycle - instruction  $N_3$  with the code 0011;
- in the 6th cycle - instruction  $N_4$  with the code 1010;
- in the 7th cycle - instruction  $N_5$  with the code 1001;
- in the 9th cycle - instruction  $N_6$  with the code 1100;
- in the 11th cycle - instruction  $N_7$  with the code 0110;
- in the 12th cycle - instruction  $N_8$  with the code 1111;
- in the 14th cycle - instruction  $N_9$  with the code 1101;
- in the 16th cycle - instruction  $N_{10}$  with the code 0111.

Since the instructions are of the four-column type, the number lines of the matrix of the unit should consist of four ferrite cores, and the output windings of the ferrite cores of the same columns should be assembled in the general readout buses of the matrix. It is convenient to take the number of lines as equal to four, since both shift registers in this case contain four transistor cells in all. Actually, during the time in which the instructions are produced, which is equal to 16 cycles, one four-cell register ensures the writing of the codes in four cycles, and a second ensures the reading of the codes in each cycle. Clock pulses of different frequencies should be fed to the cells of the register.

The circuit of the unit ensuring issuance of the instructions in the above sequence, is given in Fig.242. The circuit contains all basic components of a "hard" program system with a magnetic matrix. Each ferrite core of the matrix is symbolized in the diagram by a rectangle which is intersected by lines which represent conventional symbols for the write, read, and output buses. The write buses connected to the outputs of the first shift register are denoted by broken lines in this diagram; the read buses, connected to the outputs of the second shift register, are shown by vertical lines; and the output buses are indicated by horizontal lines.

The master oscillator MO feeds the frequency divider FD with clock pulses having a fundamental frequency  $f$ . In this case, the time in which the instructions are issued is equal to 16 cycles; therefore, the divider consists of four cascades which provide for the successive division of the fundamental frequency by 2, 4, 8, and 16. The amplifiers A are used for amplifying and shaping the code pulses produced in the output buses of the matrix.

Both shift registers of the unit are the same and consist of four ferrite-transistor cells. The output of each cell forms the output of the corresponding register which causes the output pulses to form after every frequency half- /409 period of the clock pulses feeding the given register. The outputs of the first



register are denoted by Roman numerals and those of the second register by Arabic numerals.

For the first write-in of the code "1" into the first register, pulses with a frequency of  $f/16$  are used whereas pulses with a frequency of  $f/8$  are used as the clock pulses for this register. Therefore, the signals at the outputs I - IV are generated in succession after every four cycles, within the time intervals  $t_1 - t_{1v}$  (Fig.243).

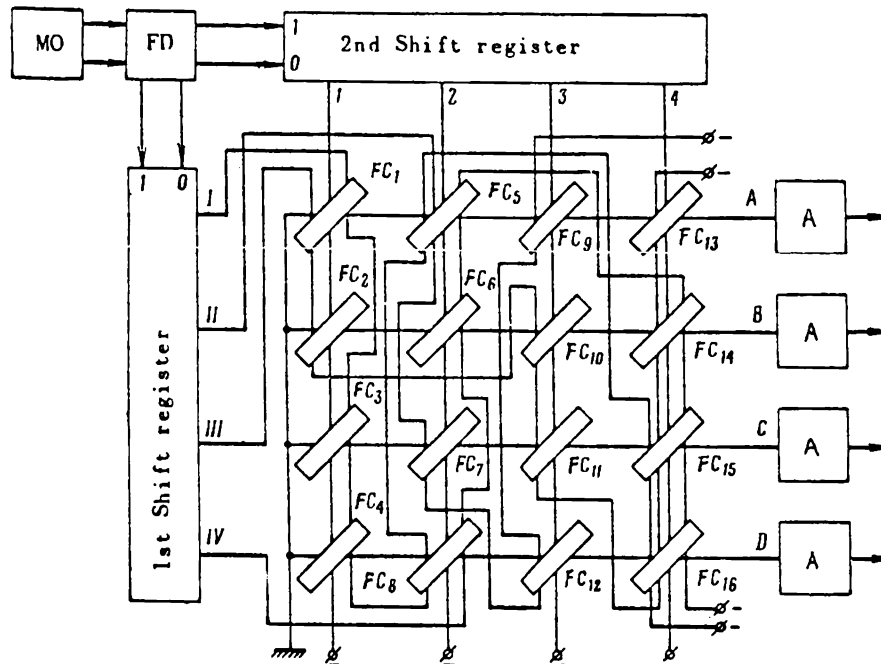


Fig.242 "Hard" Program System with Ferrite Matrix

For the first recording of the code "1" in the second register, pulses with a frequency of  $f/4$  are used while pulses with a frequency of  $f/2$  are used as the clock pulses for this register. Therefore, the signals at outlets 1 - 4 are formed consecutively after each cycle at the times  $t_1, t_2, t_3$ , etc.

The magnetic matrix consists of four number lines, each of which consists of four ferrite cores. The cores comprising one number line are arranged in Fig.242 along the vertical line. Each core of the number line represents a definite instruction code digit. Also, the cores of the upper matrix series are the most significant digits MSD of the instruction code while the cores of the lower series represent the least significant digits LSD. The matrix has four outputs, labeled in Fig.242 by A, B, C, and D. If, for example, pulses appear simultaneously at the outputs A, C, and D, this would mean that the instruction code 1011 is read from the matrix.

Readout of the instruction codes from the matrix in the required sequence

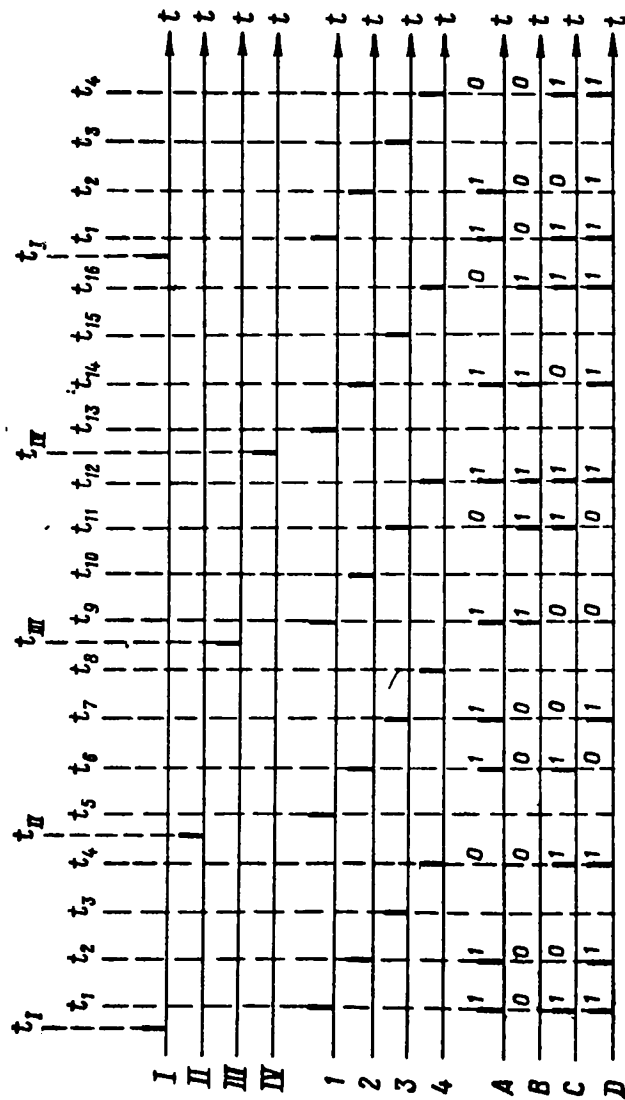


Fig.243 Timing Chart for Operation of the Unit with a Ferrite Matrix

is ensured by a determinate piercing of the matrix, i.e., by the gating of the write buses through the cores in a given sequence. Moreover, each read bus passes through all ferrite cores of the corresponding number line.

In the given case, to ensure that the instructions are issued in the required sequence, the matrix is pierced as follows: The first write bus connected to the output I of the first shift register is gated through the ferrite cores  $FC_1$ ,  $FC_3$ , and  $FC_4$  of the first number line, through the cores  $FC_5$  and  $FC_8$  of the second number line, and through the cores  $FC_{15}$  and  $FC_{16}$  of the fourth line. The fourth write bus is gated through the cores  $FC_5$  and  $FC_7$  of the second and the cores  $FC_9$  and  $FC_{12}$  of the third number lines. The third write bus is gated through the cores  $FC_1$  and  $FC_2$  of the first, the cores  $FC_{10}$  and  $FC_{11}$  of the third, and the cores  $FC_{13}$ ,  $FC_{14}$ ,  $FC_{15}$ , and  $FC_{16}$  of the fourth number lines.

The initial state of all the cores of the matrix is the zero state. At the instant  $t_1$ , at the output I of the first shift register, a pulse is produced which sets the cores  $FC_1$ ,  $FC_3$ ,  $FC_4$ ,  $FC_5$ ,  $FC_8$ ,  $FC_{15}$ , and  $FC_{16}$  to the state 1. At the instant of time  $t_1$ , i.e., in the first cycle, a pulse is generated at the output I of the second shift register. This pulse sets all cores of the first number line to the state 1; therefore, at the instant  $t_1$ , the parallel pulse code 1011 of the first instruction is produced at the outputs of the matrix.

At the instant  $t_2$ , a pulse is produced at the output 2 of the second shift register. By resetting all ferrite cores of the second number line to the zero state, this pulse generates the code 1001 of the second instruction at the output of the unit. The pulse, produced at output 3 at the instant  $t_3$ , generates a zero code since none of the cores of the third number line had been set to the state 1. The formation of the zero code corresponds to the idle time of the unit. At the instant  $t_4$ , a pulse is produced at output 4 of the second shift register. This pulse generates the code 0011 of the third instruction at the output of the unit.

After readout of the code 0011, the codes must be rewritten into the matrix since all its ferrite cores, at the instant  $t_4$ , had been reset to zero. The subsequent writing of the codes is effected by means of a pulse produced at the instant  $t_1$ , at the output II of the first shift register. This pulse sets the ferrite cores  $FC_5$ ,  $FC_7$ ,  $FC_9$ , and  $FC_{12}$  to the state 1; this ensures generation of the code 1010 of the fourth instruction at the output of the unit at the instant  $t_6$  and of the code 1001 of the fifth instruction at the instant  $t_7$ . The sub-412sequent formation of the instruction codes takes place in accordance with the timing chart (Fig.243).

This example of the construction of a "hard" program system with a magnetic matrix shows that, in cases in which the instructions or individual control pulses are issued at high frequencies, units with magnetic matrices have definite advantages over units constructed on the basis of delay lines. Actually, when magnetic matrices are used, the number of transistors in practice does not depend on the frequency at which the control pulses are generated, since they are used only in the frequency divider and shift registers.

## THE ELEMENTS OF PROGRAMMING

Section 59. The Structure of Instructions Used in Digital Computers

An instruction is a set of numbers (numeric code) defining the concrete operation which a computer must carry out. In digital computers, instructions are coded in a binary code.

In programming, instructions are recorded on a form usually in the octal system since a binary expression takes up much space and is not suited for reading purposes. When a program is fed into the computer, the octal recording is mechanically transformed into a binary recording.

As mentioned above (Section 14), depending on the particular design, computers may be of the one-, two-, three-, and four-address type and may have a natural or forced sequence of carrying out the instructions.

Let us examine in more detail the structure of the instructions used in different types of digital computers.

Three-address instruction. A three-address instruction has the form

01 0324 0245 0152.

Here, the first group of digits is the operation code; the other three groups of digits are the address.

Let 01 be the operation code of addition. Then, the given instruction has the following contents: "Take the number in cell No.0324, add it to the number stored in cell No.0245.

Enter the result of the operation - the sum of both numbers - in cell No.0152".

In comparison with other instructions, a three-address instruction more often corresponds to the logic used in carrying out an arithmetic or logical operation: Two initial numbers are taken, some operation is performed on them which results in the obtainment of a third number. When such instructions are used, most of the operations are carried out uniformly, leading to a considerable simplification of the composition of the program and the control of the operation of the computer.

Three-address computers follow a natural sequence of carrying out the /414 instructions. Our three-address computers include the large universal machines "Strela", BESM, BESM-2.

One-address instruction. A one-address instruction contains an operation code and one address.

The peculiarity of one-address computers consists in the fact that the result of the operation effected is left in the adder of the arithmetic unit.

In order to carry out the operation of addition considered above, three one-address instructions of the following contents are required:

02 0324.

The command means: "Transfer the number stored in cell No.0324 to the adder" (02 - the operation code of the transfer of the number from the memory cell into the adder);

01 0245.

This command means: "Add the number entered in cell No.0245 to the contents of the adder, leave the results of the addition in the adder" (01 - the operation code of addition):

16 0152.

This command means: "Transfer the contents of the adder to cell No.0152" (16 - the operation code of the transfer of the number from the register of the adder into the memory cell of the computer).

In order to carry out these three one-address instructions, three operating cycles of the computer are necessary. It is natural that the program of the same problem in a one-address computer must contain a larger number of instructions than in a three-address computer. However, during the solution of many problems, the difference in the volumes of the program is not too great because of the application of this method of solution with which the result of the previous operation, which remains in the adder, is used as the initial number for the subsequent operation. In addition, one-address computers have a simpler design than other types. All this determines the application of a one-address system of instructions in many modern general-purpose and special-purpose computers.

One-address computers can have only a natural sequence of carrying out the instructions. Our one-address computers are the "Ural" and "Ural-2".

A two-address instruction. Two-address instructions can be used in computers both with a natural and forced sequence of carrying out the instructions.

In a two-address instruction used in a computer with a natural sequence of carrying out the instructions, the addresses of both initial numbers are indicated.

The result of the operation can be referred to one of the addresses of these numbers or can be left in the adder.

For example, let the instruction be carried out:

/415

03 0157 0246.

The operation code 03 means: "Multiply the number stored in cell No.0157 by the number entered in cell No.0246; enter the product in cell No.0246". Thereupon, the second initial number which is located in cell No.0246 is erased and the result of the operation is entered in its stead.

If the initial numbers stored in cells No.0157 and 0246 must be retained, then such an instruction must be specified after which the result of the operation would remain in the adder, for example:

13 0157 0246.

After an instruction with such an operation code, the numbers contained in cells No.0157 and 0246 are multiplied and the product is left in the adder. By the next instruction which has a corresponding operation code, the contents of the adder can be transferred to the requisite memory cell.

In some computers using two-address instructions, the results of the operation are entered not in the locations of the initial numbers, but in cells specially prepared for this purpose whose addresses are constant.

Two-address computers may have both a natural and forced sequence of executing the instructions.

Two-address computers with a forced sequence of executing the instructions operate like one-address computers. In the second address, the number of the cell is indicated which stores the subsequent instruction.

Our computers of the "Minsk" type are two-address machines with a natural sequence of carrying out instructions.

A Sesquialteral-address computer. A sesquiaddress instruction is a kind of two-address instruction. It contains the first address in its entirety and the last half of the second address (two octal digits). The first address stands for the number of any memory cell; the second, for a specific number of memory cells whose numbers are predetermined (let us assume, cells with numbers from 4300 to 4377).

The advantage of a sesquiaddress system of instructions is the fact that it reduces the amount of digits necessary for the representation of the instruction.

A four-address instruction. Four-address instructions are used in computers with a forced sequence of carrying out instructions. The first three addresses have the same purpose as those in a three-address instruction. The fourth address indicates the cell in which the next instruction is stored. Thus, the instruction

means: "Add the numbers stored in cells No.0324 and 0245; enter the result in cell No.0152; moreover, execute the instruction entered in cell No.0362."

The forced sequence of executing instructions is used in certain foreign computers which have intermittent-operating memory units (delay lines, magnetic drums). Moreover, the instructions in the cells are so arranged that the next instruction is entered in the cell which immediately follows the cell containing the recording of the result of the previous operation. This makes it possible to reduce the time lost in waiting for access to the required memory cell, which increases the computational speed.

Four-address computers are more complicated than computers using instructions with a smaller number of addresses. In addition, the forced method of executing the instructions complicates the programming. Therefore, computers with a natural sequence of execution are now in wide use.

In addition to the operation code and the addresses, some computers contain, in their instructions, additional binary digits (signs) which determine the sequence of formation of the addresses and the operation regime. Thus, for example, in the "Ural" machine, the instruction has an address-change sign which can use two values: 0 or 1. Depending on the value of this sign, the address of the instruction remains unchanged or changes by a certain magnitude which had been written beforehand into a special memory unit.

In the "Strela" computer, the instruction contains a binary digit called the control sign.

The control sign does not affect the contents of the instruction.

The control sign may be equal to either 0 or 1. In the first case, after execution of a given instruction, the machine passes to the subsequent instruction; in the second case, it stops after execution of the instruction.

Control signs are used during operation of the computer in a special regime known as the regime of control stops. This regime is used for the program check. When operating in this regime, the computer stops after execution of each instruction which has the control sign 1.

#### Section 60. Certain Facts on the General-Purpose Computer, Necessary for Programming

The program of the solution of a given problem, before it is fed to the computer, is entered on a form in the octal system of notation.

A three-address instruction on such a form is entered by means of an octal number, e.g.,

05 0371 3254 1011.

The operation program before its input into the computer is transferred /41 from the form to punched cards or punched tape. Each octal digit entering the composition of the address or operation code is expressed by a three-digit binary number.

Thus, in the memory cell of the computer, the instruction is entered in the form of a specific set of binary digits. Therefore, the same operations can be performed on the instructions as on the binary numbers.

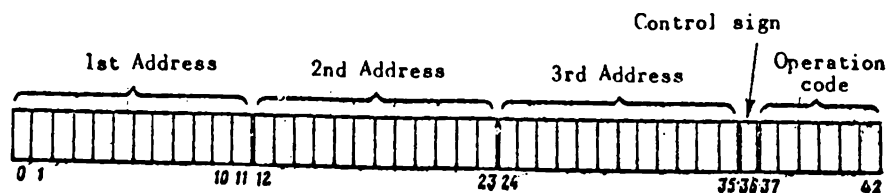


Fig.244 Distribution of the Locations in the Memory Cells of the General-Purpose Computer "Strela" during Storage of an Instruction

As an example, Fig.244 shows the distribution scheme for the locations of the memory cell in the "Strela" general-purpose computer during storage of an instruction. For writing the addresses, 36 locations are extracted, from the zeroth to the 35<sup>th</sup>. Each address occupies 12 bit locations. The control sign is stored in the 36<sup>th</sup> location. The operation code is entered in the last six locations (from the 37<sup>th</sup> to the 42<sup>nd</sup>). The memory cells are numbered in succession, beginning with zero. Let us assume that the number zero is kept constant in the cell No.0.

The memory of the computer is built up on the substitution principle: when any number is fed to the memory cell, the preceding write-in is automatically erased. If a "zero" instruction is written into the cell (a set of zeros), the computer executes no instruction and, after the cell is searched, it proceeds to the next instruction.

In addition to the working memory, there is a special memory unit in the hardware of each computer in which the programs of calculations of certain frequently encountered functions ( $\sin x$ ,  $\cos x$ ,  $\sqrt{x}$ , and others) are permanently stored.

These programs are called standard subprograms. The conversion to a required standard subprogram is effected by a special instruction. When the basic program has been executed, the computer, confronted with this instruction, automatically transmits the control to the corresponding standard subprogram. After the required function has been calculated, the execution of the basic program continues.

Certain computers such as, for example, the Strela, have a special internal



memory to store the standard subprograms. This unit has a limited capacity and cannot store a large number of standard subprograms. Therefore, standard subprograms are usually stored on magnetic drums which become part of the external memory units. /418

In most modern computers, during performance of specific operations, the arithmetic unit produces a so-called  $\omega$  (omega) signal which is used as an indication of the result (see Section 14). For example, the  $\omega$  signal is generated during addition or subtraction when the result of the operation is negative or when two numbers are compared in case of their non-coincidence. If in a given operation the  $\omega$  signal is produced, it can be said that  $\omega = 1$ ; if there is no  $\omega$  signal, then it is said that  $\omega = 0$ .

The  $\omega$  signal is fed to the control unit of the computer which, in accordance with the meaning of this signal, proceeds to some other part of the program according to special transfer instructions. The peculiarity of such transfer or jump instructions consists in the fact that they contain the addresses of the cells that store the instructions pertaining to the various portions of the program. Depending on the value of the  $\omega$  signal, one of the two commands indicated in the addresses of the jump instructions is executed. When  $\omega = 0$ , the command is executed which is stored in the cell of the first address; when  $\omega = 1$ , the command stored in the cell of the second address is executed. The  $\omega$  signal increases the logic potentialities of the computer and permits automatic changing of the sequence of executing the instructions with respect to the intermediate computational results.

In addition to the  $\omega$  signal, the arithmetic unit of most computers can feed a  $\phi$  (phi) signal to the control unit, which is produced in the case of overflow of the location network. Like the  $\omega$  signal, the  $\phi$  signal can assume two values: 0 or 1. If  $\phi = 1$ , the control unit halts the machine.

## Section 61. The Instruction System

A program-controlled digital computer can execute a strictly limited number of different operations which is called the operating system of the given computer.

The instruction system of modern general-purpose digital computers is of considerable size and contains scores of different instructions. To explain the principles of programming, it is sufficient to select from this system only the basic and most characteristic instructions.

The basic programming principles are the same for all general-purpose computers. Later, we shall examine the programming elements applicable to a three-address computer with natural sequence of executing instructions, which operates with numbers represented in normal form. This is because the programming problems for such computers are solved in an extremely simple manner. /419

To be specific, let us assume that a complete instruction code in the memory cell occupies 42 bit locations, six of which are extracted for writing

the operation code and twelve others for each address.

When an instruction is written on the forms, each set of three binary digits is expressed by one octal digit. Moreover, the operation code will be entered as two-valued, and each address as four-valued octal numbers.

In the study of an instruction system, it is convenient to use writing of the instruction in the form of letters. A three-address instruction in alphabetic writing has the form

$$abc\bar{K},$$

where  $a$ ,  $b$ , and  $c$  are the addresses of the instruction and  $K$  is the operation code.

In different makes of computers, operations identical in contents are coded in different numbers. Therefore, to designate the operation codes, we will not use numbers but conditional symbols (letters). For example, we will denote the operation code of addition by the letter  $A$ , the multiplication operation by the letter  $M$ , etc. To write the addresses, we will use both letter and numerical denotations.

Also, let us introduce certain conditional notations which we will use later:

$(a)$  = contents of the cell lettered  $(a)$ ;

$p(a)$  = exponent of the number stored in the cell lettered " $a$ ";

$m(a)$  = mantissa of the number stored in the cell lettered " $a$ ".

By the ordinal number of the instruction, we mean the running number of its storage in the memory cell.

All digital designations of the cell numbers (addresses of the instructions) are given in the octal system as is generally done in programming.

Let us now consider the characteristic operations carried out by modern digital computers.

The basic instructions of arithmetic and logical operations are given in Table 25.

### Explanation of Certain Instructions of Arithmetic and Logic Operations

1. Arithmetic operations are used to carry out arithmetic activities.

The results of the arithmetic operations are rounded off and normalized. Most machines provide for the possibility of performing these operations without rounding off or normalizing (the so-called rounding off interlock or

TABLE 25

## INSTRUCTIONS OF ARITHMETIC AND LOGICAL OPERATIONS

Conditional Designation of Operation	Type of Operation and Writing of the Instruction in General Form	Condition of Formation of the Signal $\omega = 1$	Action of the Computer after a Given Instruction
A	Addition abcA	$(c) < 0$	Algebraic addition of the numbers (a) and (b). The result of the addition is normalized and sent to cell c.
S	Subtraction abcS	$(c) < 0$	With consideration of their signs, the number (b) is subtracted from the number (a). The result of the subtraction is normalized and sent to cell c.
S <sub>1</sub>	Subtraction of the moduli abcS <sub>1</sub>	$(c) < 0$	The absolute value of the number (b) is taken from the absolute value of the number (a). The result is sent to cell c.
M	Multiplication abcM	-	Multiplication of the numbers (a) and (b) with consideration of their signs. The result is sent to cell c.
D	Division abcD	-	The number (a) is divided by the number (b) with consideration of their signs. The result is sent to cell c.
F	Formation abcF	$(c) = 0$	Digital logical addition of the numbers (a) and (b). The results are entered in the locations of the respective cell c.
C	Comparison abcC	the contents of even one	Digital execution of the operation of equivalence negation (digit compari-

Conditional Designation of Operation	Type of Operation and Writing of the Instruction in General Form	Condition of Formation of the Signal $w = 1$	Action of the Computer after a Given Instruction
		location of the cell c is not equal to zero	son). The results are entered in the locations of the respective cell c.
SA	Special Addition abcSA	-	The addresses of the instruction stored in cell a are combined with addresses of the respective cell b. The carries in the addition of the older locations of the respective addresses, are lost. The operation code entered in cell a does not change. The results of the addition of the addresses are entered in cell c.
SS	Special Subtraction abcSS	-	The respective address of the number stored in cell b is subtracted from each instruction address entered in cell a. If borrowing is necessary during the subtraction of the older locations of the respective addresses this occurs without any change of the younger location of the adjacent address at the left.

normalizing interlock) for which special operation codes are used.

In many cases, arithmetic operations can be used not in accordance with /421 their true purpose.

The operation of addition can be used to convey and set up the sign of the number. For example, on the instruction

a 0000 c A

the number (a) is added to zero (zero is always stored in cell No.0000) and will be transferred into the cell c without any change. If the number (a) is negative, the signal  $w = 1$ , indicating the sign of the number, will also be generated.

The subtraction operation is used to compare two numbers with respect to their algebraic value; from the value of the signal  $w$ , it can then be determined which of these is greater.

The operations of addition and subtraction can be used to isolate the whole or fraction part of a number. For example, let a certain number  $x$  be stored in the memory cell represented in the form of

$$x = m_x 2^p,$$

where  $m_x$  is the mantissa of the number  $x$  and  $p$  its binary exponent.

Let us assume that to express the mantissa,  $n$  bit locations are extracted. The whole part  $[x]_w$  of the number  $x$  can be isolated by the operation of addition of this number to the auxiliary number whose mantissa is equal to zero, and whose binary exponent is equal to  $n$ . The addition should be effected with the interlock of the rounding-off operation.

When the exponents are equated during performance of the addition process, the exponent  $p$  of the number  $x$  increases to the magnitude  $n$ ; at the same time, the mantissa  $m_x$  moves to the right by  $n-p$  places. The magnitude  $p$  is equal to the number of places of the mantissa in front of the decimal point. These places comprise the whole part of the number  $x$ ; the remaining  $n-p$  digits of the mantissa represent the fraction part. Therefore, when the mantissa moves to the right by  $n-p$  places, the fraction part of the number  $x$  leaves the limits of the location network and is lost. After addition and normalization, the whole part of the number  $x$  returns to its location. To exclude the possibility of distorting the whole part due to rounding off, the addition should be effected with the interlock of the rounding-off process.

To obtain the fractional part  $[x]_{f,ac}$  from the number  $x$ , its whole part  $[x]_w$  must be calculated.

For example, let the number  $x$  be stored in cell No.0100 and let an auxiliary number of the form  $0 \cdot 2^n$  be entered in cell No.0250. Let us assume that the whole part of the number  $x$  must be put in cell No.0210, and the fractional part in cell No.0211. The isolation of the whole and fraction parts is carried out with the aid of the following two instructions:

```
0100  0250  0210  A ... Addition with the interlock of the
                                rounding-off process (0210) =  $[x]_w$ 
0100  0210  0211  S ... (0211) = (0100) - (0210) =  $[x]_{f,ac}$ 
```

2. Operation F (forming) is carried out by the digital logical addition /422 of the numbers indicated in the first two addresses of the instruction. The

given operation makes it possible to form a new number or a new instruction from parts of the two sets of digits stored in the various cells.

In cell a, let the following set of digits be stored:

101101110111 000000000000 000000000000 000000,

and in cell b, the set of digits:

000000000000 000000000000 110110111111 000000.

Then, after the execution of the instruction

*abc F*

the following set of digits will be entered in cell c which is indicated in the third address:

101101110111 000000000000 110110111111 000000.

It can be easily seen that, when any number is formed with a set of zeros, the same number is obtained as the result. This feature of the formation operation is often used in subprogramming for the transfer of a number or instruction from one cell to another.

Let us assume that it is necessary to transfer a number from cell a to cell c. Since zero is always entered in cell No.0000, the instruction for the transfer of the number (a) to the cell c will look like this:

*a 0000 c F*

3. Operation C (comparison) is the digital execution of the operation of the negation of equivalence in numbers entered in the cells which are indicated in the first and second addresses of the instruction. The result of the operation, which can be determined from the third address, will have the number one in those locations where there was no coincidence.

As a rule, the operation of comparison is carried out to establish the coincidence ( $w = 0$ ) or non-coincidence ( $w = 1$ ) of the numbers being compared. The numerical result of this operation is usually not necessary for further use. The transfer instruction most often follows the operation of comparison which makes it possible, depending on the results of the comparison (value of the signal  $w$ ) to jump from one part of the program to another.

The instruction to carry out the operation of comparison has the following form:

*ab0 C*

The signal  $w = 1$  is generated upon this instruction, if the numbers in cells a and b do not coincide; or the signal  $w = 0$  is generated if the numbers do coincide. A zero is indicated in the third address since the zero cell does not receive any write-in so that the result of the operation is nowhere entered.

When necessary, the numerical value of the result can be entered in the /423 cell whose number is indicated in the third address.

4. Operations SA and SS. These operations are used to form new instructions by address modification. The addresses of the commands are modified by the addition to them (or the subtraction from them) of certain auxiliary numbers. Moreover, the operation code remains as before.

Operation SA - special addition - is used for address substitution by adding the auxiliary number. Let the following instruction be stored in cell a

*bcd M*

and let it be necessary to increase its first address by one unit, the second by two units, and the third by four units.

For this, it is necessary to use the auxiliary number which is a conditional instruction having a one in the first address, a two in the second, a four in the third, and any number (for example zero) in the operation code:

0001 0002 0004 00.

Let us feed this number to the cell e.

As a result of execution of the instruction

*afh SA*

a new instruction of the following form will be entered in cell h:

$b + 1 \quad c + 2 \quad d + 4M.$

Operation SS - special subtraction - is used for address modification by subtracting the auxiliary number.

If the following instruction is contained in cell a:

*bcd M,*

then in order to reduce its first address by one and the second by five, auxiliary numbers must be used having a one in the first address, a five in the second, and zero in the third:

0001 0005 0000 00.

Let us assume that this number is stored in cell e, and a new instruction is to be entered in cell h. Then, the instruction for address modification will have the following form:

*ae h SS*

After its execution, the following instruction will be placed in cell h:

*b - 1 c - 5 d M*

Further, let us assume that the auxiliary numbers used for address substitution can be conditionally designated as: /42/

- 1 (I) - a number having a one in the first address;
- 1 (I, III) - a number having a one in the first and third addresses;
- 2 (III) - a number having a two in the third address, etc.

By using address modification, it is often possible to considerably reduce the number of instructions entering the program in comparison with the number of instructions that can actually be executed by the computer.

Address modification, therefore, is widely used in programming.

Table 26 shows only those operations which can be performed from standard programs used in the examples considered below. There are many more such operations in general-purpose digital computers.

#### Explanation of Operations Performable with Standard Programs

The initial data are entered in the memory cell of the computer in the form of binary-coded decimal numbers. Before the calculations are started, they must be converted into a binary system. This is done with a special standard program which can be effected with the operation 10 → 2.

The computational results are obtained in binary code. In order to take them out of the computer, they are at first converted into binary-coded decimal numbers by the operation 2 → 10. After this, the binary-coded decimal number is converted by the output unit into a decimal number. /425/

Operations with standard programs can be carried out on a group of  $n$  numbers arranged in succession in the cells. For this purpose, the number  $n$ , instead of 0, should be entered in the second address of the corresponding instruction. Then, the given operation will be carried out on  $n$  numbers arranged in the cells  $a, a + 1, a + 2, \dots, a + n - 1$ . The results of the operation will be recorded in the cells  $c, c + 1, c + 2, \dots, c + n - 1$ . Let us explain the above statements on an example.



TABLE 26

INSTRUCTIONS FOR OPERATIONS WHICH CAN BE PERFORMED  
WITH STANDARD PROGRAMS

Conditional Designation of the Operation	Type of Operation and Write-In of the Instruction in General Form	Action of Computer after Instruction
$10 \rightarrow 2$	Conversion of the number from binary-coded decimal system to binary $a0c\ 10 \rightarrow 2$	The binary-coded decimal number (a) is converted to a binary number and the result is entered in cell c
$2 \rightarrow 10$	Conversion of the number from binary system to binary-coded decimal system $a0c\ 2 \rightarrow 10$	The binary number (a) is converted to a binary-coded decimal number and the result is entered in cell c.
$\sqrt{x}$	Taking the square root $a0c\ \sqrt{x}$	The machine calculates the quantity $\sqrt{(a)}$ and records it in cell c
$\sin x$	Calculation of the sine $a0c\ \sin x$	The value is determined $(c) = \sin (a)$

Let us assume that the final calculation results are distributed over the cells a, a + 1, a + 2, ..., a + 10. To take them out of the computer, it is necessary to convert them first into a binary-coded decimal system of notation. The conversion will be effected upon the instruction:

$$a\ 0011\ c\ 2 \rightarrow 10,$$

where 0011 is the octal number nine.

After execution of this instruction, the numbers from the cells a, a + 1, a + 2, ..., a + 10 will be recorded in binary-coded decimal writing and entered in the cells c, c + 1, c + 2, ..., c + 10.

In those cases where the initial numbers are not necessary for further use, the results of the operation on them can be entered in their place. Thus, in the previous example, upon the instruction

the numbers in the cells  $a$ ,  $a + 1$ ,  $a + 2$ , ...,  $a + 10$  will be converted into a binary-coded decimal system and entered in those same cells.

The basic control commands are given in Table 27.

TABLE 27  
CONTROL COMMANDS

Conditional Designation of the Operation	Type of Operation and Write-In of Instruction in General Form	Action of Computer after Instruction
CJ	Conditional jump  abc CJ	Depending on the value of the signal $w$ obtained after execution of the instruction, the control is transmitted either to the instruction recorded in cell $a$ or the instruction recorded in cell $b$ . If the signal $w = 0$ was obtained, the instruction (a) is executed; if $w = 1$ , the instruction (b) is executed. Zero is entered in cell $c$ .
Output	Output of the numbers from the computer memory for printing  a n 0000 output	$n$ numbers from the memory cells $a$ , $a + 1$ , $a + 2$ , ..., $a + n - 1$ are transferred for printing.
K	Machine stop  000 K	Machine stops

#### Explanation of the Conditional Jump Instruction

/426

The conditional jump instruction differs from all other instructions by the fact that the numbers of the cells storing the subsequent instructions are indicated in its addresses. Thus, after checking the conditional jump instruction, the computer carries the next instruction in a forced sequence.

In accordance with the value of the signal  $w$  obtained from the previous operation, the computer independently begins to carry out the program from one of the two commands indicated in the instruction addresses of the conditional jump. In the same way, the automatic operation of the computer is realized in performance of the programs providing for different directions of the solution with respect to the intermediate results.

Let us assume that, in accordance with the results of the operation of comparison of the two numbers (operation C), it is necessary to execute one of the two parts of a given program: One part begins with cell  $k$ , the other with cell  $d$ . This jump can be effected upon the conditional jump instruction

kdm CJ

If the numbers being compared coincide, the signal  $w = 0$  is produced upon which the computer begins to carry out the program, beginning with the command indicated in the first address of the conditional jump instruction (in our case, this command is  $k$ ). When the numbers being compared do not coincide, the signal  $w = 1$  is generated upon which the computer begins to execute the program from the command indicated in the second address of the conditional jump instruction (command  $d$ ). In both cases, zero is entered in the cell indicated in the third address (cell  $m$ ).

If the number of the cell in which this number is entered is indicated in the third address of the conditional jump instruction:

(a) = kda CJ,

then zero will be entered in its place after the jump. In these cases, it is said that the conditional jump instruction canceled itself.

This peculiarity of the conditional jump instruction is used when it is necessary to repeat a calculation on the portion of a program which includes the jump instruction, but when it is not necessary to effect a jump during the second calculation.

Sometimes, independent of the intermediate result, it is necessary to proceed to another part of the program which, for example, begins with the command  $b$ . This is then indicated in the first and second addresses of the conditional jump instruction:

bb0 CJ

This is called an unconditional jump instruction. Thereafter, the control is transmitted to the command  $b$  for any value of the signal  $w$ .

## Section 62. Procedure for Preparation and Solution of Problems on the Machine

//427

The preparation and solution of a problem by means of a digital computer

involve the following stages:

1. Mathematical formulation of the problem. In this stage, the mathematical functions with respect to which the calculations will be performed are conclusively established.

2. Selection of numerical method of solution. In this stage, a numerical method is selected such that the required mathematical relations can be solved with the prescribed accuracy within an acceptable interval of time, using a program of minimum volume.

3. Compilation of the working program of the machine (programming).

4. Preparation of program and numerical material for loading into the computer and the actual loading.

5. Adjustment of the program, i.e., checkup of its proper functioning, detection and correction of errors in the program, in the source data, and in the computing procedure.

6. Solution of the problem, verification of the solution, and extraction of the results.

The first two stages constitute the purely mathematical aspect of the problem, and therefore will not be considered here. The nature of the other stages will be examined below, with special attention to programming problems.

The operating program is usually compiled in accordance with the following procedure:

Compilation of the computational scheme - sequence of implementation of arithmetic and logic operations, taking into account all possible variants that may apply in the course of the computations;  
determination of the method of verifying the calculations;  
construction of the logic diagram of the program;  
compilation of the program in terms of conditional addresses;  
compilation of the program in terms of true addresses.

The computational scheme depends on the selected numerical method of solution of the problem and is laid out individually for each individual case.

The methods of verifying the computations will be examined below, along with general problems of verifying the solution of the problem.

We will begin our study of programming principles with the compilation of the logic diagrams of programs.

### Section 63. Construction of Logic Diagrams of the Program

The compilation of a program is a complex and laborious project. To facilitate programming, the process of problem-solving is subdivided into a series

of stages and a particular program is compiled for each stage. Then, all the 428 particular programs are combined into the total program.

Suppose, for example, that a program has to be compiled for computing the roots of the quadratic equation

$$ax^2 + bx + c = 0.$$

The roots of this equation are found from the formula

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

The computational procedure primarily depends on whether the radicand of the expression  $D = b^2 - 4ac$  is positive or negative. If  $D \geq 0$ , the roots are real; if  $D < 0$ , they are imaginary.

The process of solution of this problem may be broken down into the following stages:

1. Calculation of  $D = b^2 - 4ac$ .
2. Verification of the condition  $D \geq 0$  or  $D < 0$ .
3. Calculation of the real roots, if  $D \geq 0$ .
4. Calculation of the imaginary roots, if  $D < 0$ .

Clearly, the second stage should be followed by implementing either the third stage (if  $D \geq 0$ ) or the fourth (if  $D < 0$ ).

The process of solution considered above includes the arithmetic stages (1st, 2nd, 4th) and the logic stage (3rd).

The arithmetic stage consists in performing calculations according to the selected formulas. The logic stage consists in verifying satisfaction of the conditions that determine the instant of transition to the program section at which calculation from the prescribed formula commences.

In addition to the stages directly pertaining to calculation, the computational process involves other stages, given by the specific operating features of the machine, namely:

Feeding of source data - the stage of transfer of numerical data to the machine memory;  
conversion of source data to the binary system - arithmetic stage;  
conversion of results from the binary to the decimal system - arithmetic stage;  
readout of the results - stage of transfer of numerical data printers, punch cards, or other information carriers;  
stopping of machine after solution and extraction of the findings are completed.

For each stage, a separate specific program is completed. These separate programs should be implemented in a strictly definite sequence corresponding to the selected method of solution.

The diagram indicating the sequence of implementation of the individual stages during the solution of a given problem is known as the logic diagram of the program. The logic diagram of the program has the following advantages which greatly facilitate programming: /429

1) The logic diagram graphically reflects the entire process of solution of the problem by the computer.

2) The process of compiling the program is broken down into more or less autonomous parts, thus facilitating the program compilation and reducing the number of programming errors.

3) On the basis of a detailed logic diagram, a single problem can be simultaneously programmed by several programmers; the complex stages may be programmed by expert programmers and the simple stages, by technicians with lower skills.

4) The post-compilation check of the program is facilitated.

The logic diagrams of programs may be presented by two methods:

- in the form of flow charts;
- in the form of conditional symbols-operators.

Flow chart of program. In this method, the logic diagram of a program is presented in the form of rectangular blocks, each corresponding to a specific stage of solution of the problem. Within each such block, the content of the stage is concisely indicated. The blocks are linked by arrows indicating the origin and destination of the control that is being transferred in the course of the computations from block to block. A specific ordinal number is assigned to each block.

Figure 245 shows the flow chart of a program for computing the roots of a quadratic equation. In this chart, blocks 1 and 8 correspond to carry stages, block 4 is a logic stage, and the other blocks (2, 3, 5, 6, 7) correspond to the arithmetic stages. As is evident from this diagram, there exist no direct connections between blocks 5 and 6.

Operator writing of logic diagram of the program. When the logic diagram of a program is given in operator form, each stage of the computational process is presented not in the form of blocks but in that of symbols-operators. Analogous operators correspond to the arithmetic, logic and other stages. /430

Each operator is denoted by a specific symbol (letter). The following conventional symbols have been adopted for the operators:

- A - Arithmetic operator;
- P - Logic operator;

C - Carry operator;  
B - Breakpoint operator.

In addition to these operators, a series of other operators is used in programming. For example, the following operators are frequently encountered:

F(i) - Address substitution operator;  
O(i) - Renewal operator;  
‡ - Formation operator, etc.

We will consider the purpose of the latter operators in every individual example of program compilation.

The notation of the logic diagram of a program in the operator form is based on the following rules:

- 1) All operators are written in a single row.
- 2) Each operator is marked by a subscript indicating its ordinal number; the numeration is continuous.
- 3) If the operator symbols are in series, this means that the operator on the left transfers control to the operator on the right.

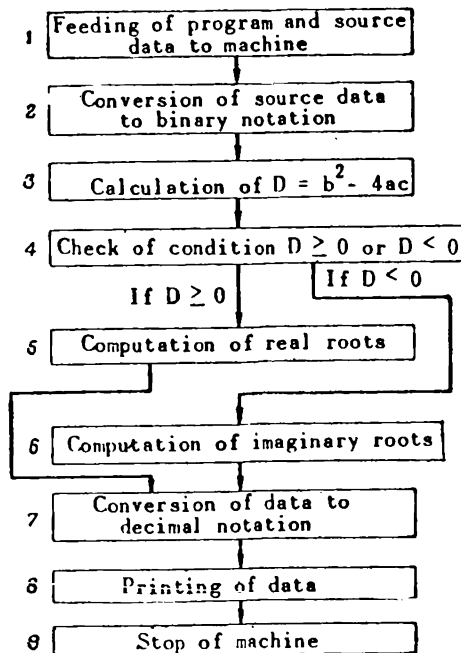
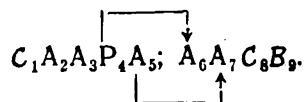


Fig.245 Flow Chart of a Program for Computing the Roots of a Quadratic Equation

4) If the operator on the left does not transfer control to its neighbor operator on the right, a radix point is placed between them.

5) The transfer of control to an operator not contiguous to its neighbor is denoted by an arrow.

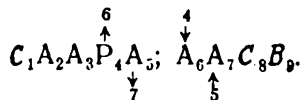
The previously considered logic diagram of the program for the solution of a quadratic equation will have the following notation in operator form:



where

- $C_1$  = carry operator for transferring the program and source data to the machine memory;
- $A_2$  = operator for converting source data to the binary system;
- $A_3$  = operator for computing  $D = b^2 - 4ac$ ;
- $P_4$  = logic operator verifying the condition  $D \geq 0$  or  $D < 0$ ;
- $A_5$  = operator for computing the real roots;
- $A_6$  = operator for computing the imaginary roots;
- $A_7$  = operator for converting the results to decimal notation;
- $C_8$  = operator for outputting the results to the printer;
- $B_9$  = breakpoint operator.

If the logic diagram of a program occupies two or more rows or involves 431 numerous transfers of control, it is inconvenient to employ long arrows to indicate these transfers. In such cases, short arrows are used. Then the logic diagram described above will read:



The arrow pointing away from an operator denotes transfer of control by that operator. The arrow pointing toward an operator denotes reception of control by that operator. The arrows are labeled with numerals indicating the numbers of the operators to which control is being transferred or from which it is being received.

The arrows denoting transfer of control with the signal  $\omega = 1$  are given above the row, while those with the signal  $\omega = 0$  are given below the row. In unconditional transfer of control, the position of the arrow above or below the row is arbitrary.

The operator, as a section of the program, must satisfy the following requirements:

1) It must realize operations of one kind, e.g., perform arithmetic calculations or verify a logic condition.



2) Control from other operators is received only by the first instruction of the operator.

3) Within the operator, control is transferred from one instruction to another in accordance with the sequence of instruction numbers.

4) Control is transferred to the next operator only from the last instruction of the operator concerned.

The logic diagram of a program is recorded more compactly in operator form than in the form of a flow chart. On the other hand, flow charting of a program is more graphic and easier to read.

In programming, both these methods of writing the logic diagram of programs are used.

Below, we will often use the operator form of writing, since this saves space; the programs to be discussed will be fairly elementary.

#### Section 64. Compilation of Programs in Conditional and Real Addresses

In the course of programming two types of operations must be performed:

Distribute the memory cells between the material pertaining to the solution of the problem (program instructions, source data, auxiliary constants, intermediate and final results of computations).  
Compile the program.

Both operations are interrelated. The program cannot be prepared unless/432 the addresses of the source numbers and auxiliary numbers as well as the addresses of the intermediate and final results are known. Conversely, the memory locations cannot be assigned unless the amount of program instructions is known in advance.

To resolve this difficulty, compilation of programs in terms of conditional addresses is used, where the cell addresses are denoted by alphanumeric symbols.

When a program is assembled in terms of conditional addresses, the machine memory is divided into two parts: one for recording the program instructions, and the other for recording the source data and the intermediate and final results. The latter part, in turn, is subdivided into three groups of locations: cells for source and auxiliary numbers, cells for intermediate results (working cells), and cells for final results.

Each group of cells is denoted by its own alphanumeric symbol.

Below, we will adopt the convention of using the following alphanumeric symbols to denote the cells in terms of conditional addresses:

- the ordinal numbers of cells in which the program instructions are

- placed:  $a + 1, a + 2, a + 3, \dots$ ;
- the numbers of cells with source data:  $b + 1, b + 2, b + 3, \dots$ ;
- the numbers of cells for auxiliary constants:  $e + 1, e + 2, e + 3, \dots$ ;
- the numbers of working cells:  $c + 1, c + 2, c + 3, \dots$ ;
- the numbers of cells for recording results of the solution:  $d + 1, d + 2, d + 3, \dots$ .

The above system of alphanumeric symbols may be arbitrarily expanded as the need arises. The digital parts of the conditional addresses are written in the octal system of notation.

After the program is assembled in conditional addresses, it will be arranged in terms of real addresses. To this end, specific numerical values are assigned to the letters  $a, b, c, e$ , and  $d$ . After this, all instructions and source data are written on a special blank from which they are transferred to punch cards (or punch tape) and thereupon fed to the machine.

Let us consider an example of assembling a program for the solution of a simple formula dependence.

Example. Let us assemble a program for the computation of  $y$  based on the formula

$$y = \frac{x^3 - Ax^2 + Bx}{Cx^2 - 1}.$$

The logic diagram of the program will have the form

$$A_1 A_2 A_3 C_4 B_5,$$

where

- $A_1$  = operator for conversion of source data to the binary system;
- $A_2$  = arithmetic operator for computing  $y$ ;
- $A_3$  = binary-to-decimal conversion operator;
- $C_4$  = operator for outputting the results to the printer;
- $B_5$  = breakpoint operator.

/433

Before the work can be started, the program and the source data must be inserted in the machine. This may be done manually from a console, but in practice automatic insertion based on a special input program is used. The nature of the program is largely determined by the design of the computer, and the program itself is of no special interest to the study of programming principles.

Let us place the source data in the following cells (see Table).

We will set aside the cells with the ordinal numbers  $a + 1, a + 2, a + 3, \dots$  for accommodating the program. We will use the cells with the ordinal numbers  $c + 1, c + 2, c + 3, \dots$  as the working cells. The initial results will be placed in the cell  $d + 1$ .

No. of Cell	$b + 1$	$b + 2$	$b + 3$	$b + 4$	$b + 5$
Number stored in cell	$x$	$A$	$B$	$C$	1

Assembly of the program. The running numbers of program instructions are separated by a parenthesis from the instruction itself, and the content of the instruction is recorded opposite each such number:

Program Instruction	Explanations
$a + 1) \quad b + 1 \quad 5 \quad b + 1 \quad 10 \rightarrow 2$	Convert five source numbers written into cells $b + 1$ , $b + 2$ , $b + 3$ , $b + 4$ and $b + 5$ to binary notation and write them into the same cells
$a + 2) \quad b + 1 \quad b + 3 \quad c + 1 \quad M$	Determine the value of $Bx$ and write it into cell $c + 1$
$a + 3) \quad b + 1 \quad b + 1 \quad c + 2 \quad M$	Determine the value of $x^2$ and write it into cell $c + 2$
$a + 4) \quad b + 2 \quad c + 2 \quad c + 3 \quad M$	Determine the product of $Ax^2$ and write it into cell $c + 3$
$a + 5) \quad b + 4 \quad c + 2 \quad c + 4 \quad M$	Determine the value of $Cx^2$ and write it into cell $c + 4$
$a + 6) \quad b + 1 \quad c + 2 \quad c + 2 \quad M$	Determine the value of $x^3$ ; since the quantity $x^2$ is no longer needed, $x^3$ replaces it in cell $c + 2$
$a + 7) \quad c + 2 \quad c + 3 \quad c + 2 \quad S$	Determine the difference $x^3 - Ax^2$ and place it into cell $c + 2$
$a + 10) \quad c + 2 \quad c + 1 \quad c + 1 \quad A$	Find the sum of $(x^3 - Ax^2) + Bx$
$a + 11) \quad c + 4 \quad b + 5 \quad c + 2 \quad S$	Determine the difference $Cx^2 - 1$
$a + 12) \quad c + 1 \quad c + 2 \quad d + 1 \quad D$	Use this instruction to find the sought value of $y$ , and write it into the reply cell $d + 1$
$a + 13) \quad d + 1 \quad 0001 \quad d + 1 \quad 2 \rightarrow 10$	Convert the number recorded in the cell $d + 1$ (result of the calculations) to the binary decimal-coded system and rewrite it into the same cell <span style="float: right;">/434</span>
$a + 14) \quad d + 1 \quad 0001 \quad 0000 \quad Extr.$	Output the number from the machine to a printer
$a + 15) \quad 0000 \quad 0000 \quad 0000 \quad K$	Stop the machine

After assembling the conditional-address program, we will next compile the real-address form of this program.

Let us assume that the program can be placed in the computer memory beginning with the cell No.0020. Then, assuming  $a = 0017$ , we obtain specific numerical addresses for each instruction. The program will occupy the cells from

No.0020 to No.0034. Further, assuming  $b = 0034$ , we place the source data in the cells beginning with No.0035 and ending with No.0041. In the same manner, we obtain the numbers of the working cells: 0042, 0043, 0044, 0045. The result of the calculations will be placed in cell No.0046.

Now the distribution of memory cells with respect to the entire problem complex can be written in real-address notation.

Operator	No. of Cell	Instruction				Remarks
$A_1$	0020	$10 \rightarrow 2$	0035	0005	0035	Program
$A_2$	0021	M	0035	0037	0042	
	0022	M	0035	0035	0043	
	0023	M	0036	0043	0044	
	0024	M	0040	0043	0045	
	0025	M	0035	0043	0043	
	0026	S	0043	0044	0043	
	0027	A	0043	0042	0042	
	0030	S	0045	0041	0043	
	0031	D	0042	0043	0046	
$A_3$	0032	$2 \rightarrow 10$	0046	0001	0046	
$C_1$	0033	Extr.	0046	0001	0000	
$B_1$	0034	K	0046	0000	0000	
	0035		X			Source data
	0036		A			
	0037		B			
	0040		C			
	0041		1			Working cells
	0042					
	0043					
	0044					
	0045					Reply cell
	0046					

1435

## Section 65. Branching Programs

In many cases, the course of the computational process depends on the intermediate results of the calculations.

Naturally, in programming, the possible variants of the computational process must be determined in advance. A special program section must be set aside

for each of these variants.

Computational processes involving several different variants of the computational process are normally termed branching processes, and the programs which they follow are known as branching programs.

Two instructions are used to branch a program:

preliminary, determining the beginning and direction of the branching;  
implementing, determining the transition to the required program section.

The preliminary instruction serves to perform the operation required to obtain the signal  $\omega$  of the desired value. The value  $\omega = 0$  pertains to one branch of the computational process and the value  $\omega = 1$ , to the other branch. The implementing instruction is the instruction for conditional transition.

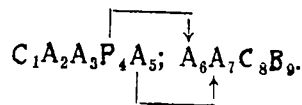
Consider an elementary example of a branching program.

/436

Example. Compile a program for the solution of the quadratic equation

$$ax^2 + bx + c = 0.$$

The logic diagram of the program for the solution of the quadratic equation was examined earlier (cf. Section 61). It has the following operator form:



Let us place the source data in the following cells:

No. of Cells . . . .	$b + 1$	$b + 2$	$b + 3$	$b + 4$
Number stored in cell .	$a$	$b$	$c$	2

We will set aside cells  $a + 1$ ,  $a + 2$ , ... for the instructions of the basic program and place the replies into cells  $d + 1$ ,  $d + 2$ , ... .

The content of the operator  $C_1$  is of no interest here, so that we will assemble the program, beginning with the operator  $A_2$ . This operator consists of a single instruction serving to ensure access to a standard subroutine for the conversion of source data to binary notation.

Let us assemble a program section consisting of the operators  $A_2$  (instruction  $a + 1$ ) and  $A_6$  (instructions  $a + 2$  to  $a + 7$ ).

Program Instruction	Explanation
$a + 1) \quad b + 1 \quad 4 \quad b + 1 \quad 10 \rightarrow 2$	Convert the source data to the binary system
$a + 2) \quad 0000 \quad b + 2 \quad c + 1 \quad S$	Derive the quantity $-b$
$a + 3) \quad b + 2 \quad b + 2 \quad c + 2 \quad M$	Derive the quantity $b^2$
$a + 4) \quad b + 1 \quad b + 4 \quad c + 3 \quad M$	Derive the quantity $2a$
$a + 5) \quad b + 4 \quad c + 3 \quad c + 4 \quad M$	Derive the quantity $4a$
$a + 6) \quad b + 3 \quad c + 4 \quad c + 4 \quad M$	Derive the quantity $4ac$
$a + 7) \quad c + 2 \quad c + 4 \quad c + 4 \quad S$	Derive the quantity $D =$ $= b^2 - 4ac$

Instruction  $a + 7$  is a preliminary instruction which determines the beginning and direction of the branching. If execution of this instruction results in  $D \geq 0$ , the signal  $\omega = 0$  will be triggered, while if it results in  $D < 0$ , the signal  $\omega = 1$  will be triggered.

The operator  $P_4$ , which transfers the control to the required branch of 43 the program, consists of a single instruction for conditional transfer or jump

$$a + 10) \quad a + 11 \quad g + 1 \quad 0000 \quad CJ.$$

At  $D \geq 0$ , the control will be transferred to the instruction  $a + 11$ ; at  $D < 0$ , it will be transferred to the instruction  $g + 1$ .

The instruction  $a + 11$  is the initial instruction for the operator  $A_6$ , which latter computes the real roots  $x_1$  and  $x_2$ ; the instruction  $g + 1$  lies at the origin of the operator  $A_6$  which determines the imaginary complex-conjugate roots

$$A + iB \text{ and } A - iB,$$

where

$$A = -\frac{b}{2a};$$

$$B = \frac{\sqrt{4ac - b^2}}{2a}.$$

Therefore, in the presence of real roots, the computer gives the reply in the form of the numbers  $x_1$  and  $x_2$ , whereas in the presence of imaginary roots the reply is the numbers  $A$  and  $B$ . To determine which case took place in the solution of a specific equation, a conditional mark, indicating which one of the two branches of the program was followed in the computation, must be fed to a cell set aside especially for this purpose. Suppose that this mark is represented by the number 2, fed to the cell  $d + 3$ , if the roots are real or by the number 4, if the roots are imaginary. We will reserve the cell  $d + 1$  for the number  $x_1$  or  $A$  and the cell  $d + 2$  for the number  $x_2$  or  $B$ .

We then assemble the first branch of the program, used for determining real

roots.

Operator A<sub>5</sub>:

Program Instruction	Explanation
a + 11) b + 4 0000 d + 3 F	Feed the number 2 - mark for real roots - to the cell d + 3
a + 12) c + 4 0000 c + 4 $\sqrt{x}$	Extract the quantity $\sqrt{b^2 - 4ac}$
a + 13) c + 1 c + 4 c + 5 A	Extract the quantity $-b + \sqrt{b^2 - 4ac}$
a + 14) c + 5 c + 3 d + 1 D	Extract the first root $x_1$
a + 15) c + 1 c + 4 c + 5 S	Extract the quantity $-b - \sqrt{b^2 - 4ac}$
a + 16) c + 5 c + 3 d + 2 D	Extract the second root $x_2$

The operators A<sub>7</sub>, C<sub>8</sub>, and B<sub>9</sub> consist of a single instruction each:

a + 17) d + 1 3 d + 1 2→10	Convert to the binary-coded decimal notation of the numbers placed in cells d + 1, d + 2, d + 3.
a + 20) d + 1 0000 0000 Extr.	Output the reply to the printer
a + 21) 0000 0000 0000 K	Stop the machine

The last three instructions, beginning with a + 17 are common to both branches. Therefore, at the end of the second branch of the program, transfer to the instruction a + 17 is required; this may be accomplished by means of an unconditional instruction.

The second branch of the program (Operator A<sub>6</sub>) will have the following 438 form:

Instruction Program	Explanation
g + 1) b + 4 b + 4 d + 3 M	Feed the number 4 - mark for imaginary roots - to cell d + 3
g + 2) 0000 c + 4 c + 4 S	Reverse the sign y of the quantity $D = b^2 - 4ac$ . The quantity $4ac - b^2$ will be found in the cell c + 4
g + 3) c + 4 0000 c + 4 $\sqrt{x}$	Extract the quantity $\sqrt{4ac - b^2}$
g + 4) c + 1 c + 3 d + 1 D	Extract the real part of the roots $A = -\frac{b}{2a}$
g + 5) c + 4 c + 3 d + 2 D	Extract the number $B = \frac{\sqrt{4ac - b^2}}{2a}$

Instruction Program	Explanation
$g + 6) \ a + 17 \ a + 17 \ 0000 \ CJ$	Unconditional transfer to the instruction $a + 17$ which initiates the end - common to both branches - of the program.

## Section 66. Cyclic Programs

Multiply-recurring program sections are often employed in computational processes. Thus, in the solution of many problems the computation must be performed with the aid of the same formulas in which, each time, new source data must be substituted.

Such computational processes are normally termed cyclic, and the multiply-recurring program sections are known as cycles.

If the number of cycles is known beforehand, a developed program in which all the cycles are incorporated one by one in sequence of their performance, may be assembled for the cyclic process. If the number of cycles is large, developed programs become too cumbersome and their assembly is extremely time-consuming. At times, they prove so large that they cannot be accommodated in the computer memory.

There exist many problems for which the number of cycles cannot be determined in advance. These include, for example, problems for which the calculation is performed by the method of successive approximations to a desired degree of accuracy. For problems of this kind it is generally impossible to assemble a developed program.

In practice, when programming cyclic computational processes, so-called cyclic programs are assembled.

Each cycle of a program of this kind consists, in the most elementary case, of three operators:

The arithmetic operator, for calculation according to a multiply used /439 formula;

The operator for substitution of new source data in the formula; normally this is a carry operator or an address substitution operator;

The logic operator, which controls the cycle; it controls the number of repetitions of the cycle, ensures the return to the origin of the cycle, and determines the moment of emergence from the cycle and transfer to further calculations.

We will examine different types of cycles.

Iteration cycle. Iteration cycles are used in programs assembled for the



solution of mathematical problems by the iterative method, i.e., by the method of multiple repetition of calculations according to the same formula.

The nature of the iterative method lies in that the sought quantity  $y$  is found by means of successive approximations, where the results of the preceding calculations are used as source data for the succeeding calculations. At first the initial rough value of this quantity  $y_0$  is derived and substituted in a given formula. Then the next approximate value  $y_1$  is determined from this formula and is substituted in this formula. As a result, the next successive approximation  $y_2$  is found, and so on. Each succeeding value  $y_{i+1}$ , compared with the preceding  $y_i$ , is closer to the true value of the sought quantity  $y$ . The calculations are continued to a desired degree of accuracy, i.e., until the difference between adjacent values  $y_{i+1}$  and  $y_i$  becomes negligibly small in absolute value.

In other words, given a certain degree  $\epsilon$  of accuracy, the calculations are continued until

$$|y_{i+1}| - |y_i| \leq \epsilon. \quad (51)$$

A feature of the iterative method of calculation is the impossibility of determining in advance the number of cycles required to obtain the result with the desired accuracy.

The logic diagram of the program for the solution of a problem by the iterative method has the following possible general form:

$$S_0 A_1 C_2 A_3 P_4 A_5 C_6 B_7.$$

$\uparrow$

where the operators  $S_0$  and  $A_1$  are the program input operator and the operator for the conversion of source data to the binary system, respectively; the operators  $C_2$ ,  $A_3$ , and  $P_4$  make up the iteration cycle of the program;  $A_5$  is the operator for conversion of the data from binary to the binary-coded decimal system;  $C_6$  is the operator for extracting the data from the machine;  $B_7$  is the breakpoint operator.

Consider the contents of the operators that make up the iteration cycle, since the purpose of the other operators is already known.

Here,  $C_2$  is the operator for the substitution of new source data in the /440 iterative formula on which the calculations are based. Usually this is the carry operator for transferring the contents of the cell  $b + 1$  to the working cell  $c + 1$ . The cell  $b + 1$  always stores the next approximation  $y_1$  of the sought quantity  $y$ . Before starting the calculations, the quantity  $y_1$  is transferred to the working cell  $c + 1$ , whence it is taken as the source quantity for calculations based on the selected iterative formula.

The quantity  $A_3$  is the arithmetic operator for calculations based on the

iterative formula. The result of the calculation of  $y_{i+1}$ , which is the next approximation to the sought quantity, is stored in cell  $b + 1$ , replacing the preceding value of  $y$ .

The quantity  $P_4$  is the logic operator which controls the cycle. This operator serves to verify satisfaction of the condition (51). If this condition is not satisfied, control is transferred to the operator  $C_2$  and the cycle is repeated. If the condition is satisfied, control is transferred to the operator  $A_5$ .

In the logic diagram described below, the arrow indicating the transfer of control from the operator  $P_4$  to the operator  $C_2$  is entered below the row which corresponds to the signal  $\omega = 0$ . If control is transferred in the presence of  $\omega = 1$  this arrow is drawn above the row.

Example 1. Assemble a program for finding the quantity  $y = \frac{1}{x}$  by the iterative method to within  $\epsilon$ .

The reciprocal is usually determined from the following iterative formula, familiar in the mathematics of calculus

$$y_{i+1} = y_i (2 - y_i x).$$

If the initial value  $x$  is presented in the form of  $x = m \cdot 2^p$ , where  $\frac{1}{2} < m < 1$ , then  $y_0 = 2^{-p}$  may be taken as the initial approximation. The quantity  $y_1$  is derived from the value of  $y_0$  in accordance with the iterative formula, after which  $y_2$  is derived from the value of  $y_1$ , and so on.

Programming of the problem must be started by laying out the logic diagram of the program. Here, we will adopt the following above-described logic diagram of the iteration cycle:

$$S_0 A_1 C_2 A_3 P_4 A_5 C_6 B_7.$$

↑

The function of each of the operators entering this diagram has been discussed above.

The given iterative equation is solved until the inequality

$$|y_{i+1} - y_i| - |\epsilon| < 0.$$

is satisfied.

Satisfaction of this condition is verified by the operator  $P_4$ .

We then store the source data of the problem in the following memory loca-

$b + 1$	$b + 2$	$b + 3$	$b + 4$
$y_0$	$x$	$\epsilon$	2

After this, we assemble the program beginning with the operator  $A_1$ . /447

Operator	No. of Instruction	Contents of Instruction	Explanation
$A_1$	$a + 1$	$b + 1$ 0004 $b + 1$ 10→2	Convert the source data to the binary system
$C_1$	$a + 2$	$b + 1$ 0000 $c + 1$ F	Transfer the initial value $y_1$ to the working cell $c + 1$
$A_2$ {	$a + 3$	$c + 1$ $b + 2$ $c + 2$ M	Find the quantity $y_1 x$
	$a + 4$	$b + 4$ $c + 2$ $c + 2$ S	Find the quantity $2 - y_1 x$
	$a + 5$	$c + 1$ $c + 2$ $b + 1$ M	Find the quantity $y_{1+1}$ and place it in cell $b + 1$
$P_1$ {	$a + 6$	$b + 1$ $c + 1$ $c + 1$ S	Find the difference $y_{1+1} - y_1$
	$a + 7$	$c + 1$ $b + 3$ 0000 $S_1$	Determine the sign of the difference $ y_{1+1} - y_1  -  \epsilon  = \Delta$
	$a + 10$	$a + 2$ $a + 11$ 0000 CJ	Conditional transfer instruction. If $\Delta \geq 0$ , so that $\omega = 0$ , transfer the control to the instruction $a + 2$ and repeat the cycle. If $\Delta < 0$ , take $y_{1+1}$ as the sought value of $y$ to within $\epsilon$
$A_3$	$a + 11$	$b + 1$ 0001 $b + 1$ 2→10	Convert the results to the binary-coded decimal system
$C_2$	$a + 12$	$b + 1$ 0001 0000 Extr.	Transfer the results from the memory to the print-out
$B_1$	$a + 13$	0000 0000 0000 K	Halt the computer

Loop with address modification. There are some problems whose solution requires multiple calculations with the same formulas and with the number of repetitions known beforehand. Thus, Tables of the values of  $y$  for different values of the argument  $x = x_1, x_2, x_3, \dots, x_n$  must frequently be computed.

In each individual case the dependence of  $y$  on  $x$  is expressed by a specific formula, such as

$$y = x^2, y = \cos x, y = \sqrt{x^2 - 4x}, \text{ etc.}$$

In the general case,  $y$  may be any function of  $x$ , i.e.,

$$y = f(x).$$

To determine the quantity  $y = f(x)$  for different values of the argument  $x$ , a cyclic or loop program is required. In such a program, to ensure the inflow of new source data to the arithmetic operator, it is suggested, before repeating the cycle, to modify the addresses of the cells that store the source data. This is done with the aid of a special operator incorporated into the loop, known as the address modification operator.

The address modification operator is denoted by the symbol  $F(i)$ , where  $i$  is the ordinal number of the loop, corresponding to the ordinal number of the argument. The quantity  $i$  is the parameter of the loop.

The logic diagram of address modification in loops has the form: /442

$$\overline{A_n^i F_{n+1}(i) P_{n+2}} \quad (\text{the arrow is drawn arbitrarily above the line}).$$

In this case  $A_n^i$  is the arithmetic operator of the loop. The superscript  $i$  near the operator symbol indicates that this operator changes with any variation in the parameter  $i$ . Variations in the operator  $A_n^i$  occur as a result of the address modification of part of its instructions. Operators of this kind are parameter-dependent. Here,  $F_{n+1}(i)$  is the address modification operator, while  $P_{n+2}$  is the loop control operator.

Example. Assemble a program for computation of the Table of the squares of a natural number series from 1 to  $n$ , using a cycle with address modification.

Let us load the source data into the following locations:

$b + 1$	$b + 2$	$b + 3$	$\dots$	$b + n - 1$	$b + n$
1	2	3	$\dots$	$n - 1$	$n$

The logic diagram of the program is

$$\overline{S_0 A_1 A_2^i F_3(i) P_4^i A_5 C_6 B_7}.$$

In this diagram, the operators  $I_0$ ,  $A_1$ ,  $A_5$ ,  $C_6$ ,  $B_7$  are the same as in the preceding example;  $A_2^i$  is the arithmetic operator for computation according to the formula  $y = x^2$ ;  $F_3(i)$  is the operator for the address modification of the instructions of the operators  $A_2^i$  and  $P_4^i$ ; and  $P_4^i$  is the logic operator controlling the loop (in this program, it is dependent on the parameter  $i$ ).

Assembling the program. Let us denote the modifiable addresses of program instructions by asterisks:

Operator	$A_1$	$a + 1)$	$b + 1$	$n$	$b + 1$	$10 \rightarrow 2$
"	$A_2^1$	$a + 2)$	$b + 1*$	$b + 1*$	$b + 1*$	$M$
"	$F_3(i)$	$\begin{cases} a + 3) \\ a + 4) \end{cases}$	$\begin{cases} a + 2 \\ a + 5 \end{cases}$	$\begin{cases} e + 1 \\ e + 2 \end{cases}$	$\begin{cases} a + 2 \\ a + 5 \end{cases}$	$\begin{cases} SA \\ SA \end{cases}$
"	$P_4^1$	$\begin{cases} a + 5) \\ a + 6) \end{cases}$	$\begin{cases} b* \\ a + 7 \end{cases}$	$\begin{cases} b + n \\ a + 2 \end{cases}$	$\begin{cases} 0000 \\ 0000 \end{cases}$	$\begin{cases} C \\ CJ \end{cases}$
"	$A_5$	$a + 7)$	$b + 1$	$n$	$b + 1$	$2 \rightarrow 10$
"	$C_6$	$a + 10)$	$b + 1$	$n$	$0000$	$Extr.$
"	$B_7$	$a + 11)$	$0000$	$0000$	$0000$	$K$

The operator  $F_3(i)$  consists of two instructions:  $a + 3$  and  $a + 4$ . The instruction  $a + 3$  is used for the address modification of all three addresses of the instruction  $a + 2$ , by means of the auxiliary number 1 (I, II, III) loaded into the cell  $e + 1$ . The instruction  $a + 4$  serves for the address modification of the instruction  $a + 5$  of the operator  $P_4$ , by means of the number  $(e + 2) = 1$  (I). The operator  $P_4$  controls the loop by comparing the content of the cell from which the operator  $A_5^1$  extracts the next initial quantity, with the content of the cell  $b + n$ . So long as the contents of these locations do not coincide, the operator  $P_4^1$  ensures repetition of the loop. After the loop has been repeated  $n$  times, the instruction  $a + 5$  assumes the following form:

$$a + 5) \ b + n \ b + n \ 0000 \ C .$$

In accordance with this instruction, the content of the cell  $b + n$  is compared with itself. This will cause a signal  $\omega = 0$  to appear, which transfers control to the operator  $A_5$ . Since the operator  $F_3(i)$  is in front of the instruction  $a + 5$ , it performs address modification until this instruction is executed. Therefore, the symbol  $b$  which, after the first address modification, will assume the form  $b + 1$ , is placed in the first address of the instruction  $a + 5$ .

Considering that, in cyclic programs with address modification, the instructions vary from loop to loop, such programs often are termed variable-cycle programs.

Loop with address modification and restoration. Problems are possible for which a variable-cycle program segment must be repeated numerous times. This happens in cases where a loop with address modification is part of another, external loop. The solution of the problem then requires restoring the original form of the instructions that had been modified in the course of performing the loop with address modification, each time before the entire external loop can be repeated.

Such return of instructions to their original form is known as instruction restoration. This may be accomplished in two ways: by means of the address modification operator or by means of the restoration operator.

The first method consists in eliminating the results of the previous address modification and the second method, in feeding the locations of the

altered instructions with sets of numbers representing these instructions in their original form.

Assume that the original form of a modified cycle instruction had been

$$a + k) b * b * c + 1 M.$$

After n-fold performance of the cycle, the instruction will assume the form

$$a + k) b + n b + n c + 1 M.$$

To restore the instruction  $a + k$  with the aid of an address modification operator, it is necessary to prestore in the auxiliary cell  $e + 1$  a set of digits with  $n$  units in the first and second addresses, i.e.,

$$(e + 1) = n(I, II).$$

Restoration of the instruction is possible by means of a SS (special subtraction) operation according to the instruction

$$a + k e + 1 a + k SS.$$

In accordance with this instruction, the computer will subtract  $n$  units from the first and second addresses of the set of digits:

$$(a + k) = b + n b + n c + 1 M,$$

as a result of which the original form of the instruction will remain in the cell  $a + k$ .

To restore the instruction  $a + k$  with the aid of the restoration operator, an auxiliary set of digits representing the original form of the instruction, must be prestored in the cell  $e + 1$ , i.e.,

$$(e + 1) = b b c + 1 M.$$

The restoration operator will transfer the content of the cell  $e + 1$  to the cell  $a + k$  which contains the instruction to be restored. Obeying the instruction

$$e + 1 0000 a + k F$$

the computer will transfer to the cell  $a + k$ , in place of the set of digits

$$b + n b + n c + 1 M$$

another set of digits

$$b b c + 1 M,$$

and thus cause restoration of the instruction.

The restoration operator is conventionally denoted by  $O(i)$ , where  $i$  is the parameter of the cycle being restored.

Depending on the method of restoration adopted in the program, logic loops with address modification and restoration are based on either of the following two schemes:

- 1) With use of the address modification operator

$$\overrightarrow{A_n^i F_{n+1}(i) P_{n+2} F_{n+3}(i)}, \quad (52)$$

where  $F_{n+3}(i)$  is the operator for address modification with respect to the parameter  $i$ , restoring the operator  $A_n^i$  to its original form;

- 2) With use of the restoration operator

$$\overrightarrow{A_n^i F_{n+1}(i) P_{n+2} O_{n+3}(i)},$$

where  $O_{n+3}(i)$  is the operator restoring the arithmetic operator  $A_n^i$  to its original form.

In both schemes the arrows are entered arbitrarily. The cycle-restoration operator itself is outside the cycle.

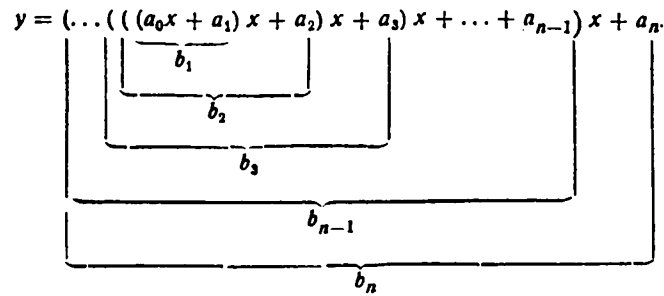
Let us consider an example of assembling a program having a loop with address modification and restoration.

Example. Assemble a program for computing the value Table of the function

$$y = a_0 x^n + a_1 x^{n-1} + a_2 x^{n-2} + \dots + a_{n-1} x + a_n$$

for the values of the argument  $x = x_1, x_2, x_3, \dots, x_n$ .

It is convenient to base the calculations on the so-called Horner diagram, where the value of  $y$  is presented as follows: /445



As can be seen from this diagram,  $y$  may be computed through multiple use of a formula of one and the same form. We will denote the sum in the first parentheses by  $b_1$ . We then find the succeeding sums with the aid of the preceding sums by means of similar arithmetic operations:

$$\begin{aligned} b_1 &= a_0x + a_1, \\ b_2 &= b_1x + a_2, \\ b_3 &= b_2x + a_3, \\ &\vdots \\ b_n &= b_{n-1}x + a_n. \end{aligned}$$

The sought value of  $y$  equals  $b_n$ . The arithmetic operator  $A$  for computing  $y$  should work according to the formula

$$b_i = b_{i-1}x_j + a_i, \quad (53)$$

where  $x_j$  is a specific value of the independent variable  $x$ .

The quantity  $y_j$  will be computed in each individual case for a specific value of the argument  $x_j$ . Since the arithmetic operator  $A$  performs calculations according to eq.(53), it should be address-modified with respect to the parameter  $i$  and denoted by the symbol  $A^i$ .

To compute the quantity  $y_{j+1}$ , which corresponds to the immediately following quantity  $x_{j+1}$  of the argument, the arithmetic operator  $A^i$  must first be restored. The new initial value  $x_{j+1}$  can be introduced into the operator  $A^i$  by two methods:

By means of the carry operator  $C^j$ , address-modified with respect to the parameter  $j$ .

By means of address modification of the operator  $A^i$  itself with respect to the parameter  $j$ ; in this case, the operator will be denoted by  $A^{ij}$ , as was done in the logic diagram (52).

Suppose the function  $y$  is a polynomial of the fourth degree:

$$y = a_0x^4 + a_1x^3 + a_2x^2 + a_3x + a_4.$$



After its transformation by the Horner method, we have

$$y = \{[(a_0x + a_1)x + a_2]x + a_3\}x + a_4.$$

We will then load the source data into the following cells:

#### POLYNOMIAL COEFFICIENT

$b$	$b + 1$	$b + 2$	$b + 3$	$b + 4$
$a_0$	$a_1$	$a_2$	$a_3$	$a_4$

#### INDEPENDENT VARIABLES

/446

$b + 5 =$ $= d + 1$	$d + 2$	$d + 3$	$\dots$	$d + j$	$\dots$	$d + n = b +$ $+ n + 4$
$x_1$	$x_2$	$x_3$	$\dots$	$x_j$	$\dots$	$x_n$

During solution of the problem, the cells  $d + 1$ ,  $d + 2$ ,  $d + 3$ , ... will be used as reply cells in which the arguments will be replaced by the corresponding values of the function  $y_1$ ,  $y_2$ ,  $y_3$ , ... .

Next, we will assemble the program. Let us base the logic diagram of the first half of the program, including the computational cycle, on eq.(53)

$$S_0 A_1 \overline{C_2^j A_3^i F_4(i) P_5 C_6^j} \dots$$

In this diagram,  $S_0$  is the program input operator,  $A_1$  is the operator for conversion of the source data to binary notation, and  $C_2^j$  is the operator for transfer of the argument  $x_j$  to the working cell  $c + 1$  and of the quantity  $a_0$  to the working cell  $c + 2$ .

Since the function  $y$  is sought for different arguments  $x_j$ , the operator  $C_2^j$  is address-modified with respect to the parameter  $j$ , for which purpose a corresponding address-modification operator should be provided in the second half of the program. Further, the operator  $C_2^j$  is followed by the arithmetic operator  $A_3^i$  which performs calculations according to eq.(53); the operator  $F_4(i)$ , which address-modifies the operator  $A_3^i$  with respect to the parameter  $i$ ; and the logic operator  $P_5$  which controls the loop. Here,  $C_6^j$  is the operator for transferring  $y_j$  from cell  $c + 2$  to cell  $d + j$ .

During determination of  $y_j$ , the loop described above operates as follows.

First the operator  $A_3^1$  computes  $b_1 = a_0 x + a_1$ , i.e., performs the following operation:

$$b_1 = (c + 2)(c + 1) + (b + 1).$$

The result  $b_1$  is written into the cell  $c + 2$ . After this, the operator  $F_4(i)$  increases the address  $b + 1$  by a one. The operator  $P_5$  repeats the cycle

$$b_2 = (c + 2)(c + 1) + (b + 2).$$

The quantity  $b_2$  again is loaded into cell  $c + 2$ , after which the operator  $F_4(i)$  increases the address  $b + 2$  by a one, and so on. The operator  $P_5$  ensures four-fold repetition of the loop, resulting in the quantity

$$b_4 = (c + 2)(c + 1) + (b + 4) = y_j.$$

We will write the first half of the program in alphanumeric form, beginning with the operator  $A_1$ :

Operator  $A_1$   $a + 1) b n + 5 b 10 \rightarrow 2$ .

This operator converts the polynomial coefficients  $a_0, a_1, a_2, a_3, a_4$  and the arguments  $x_1, x_2, x_3, \dots$  to the binary system since these quantities are arranged in sequence, beginning with the cell  $b$  and ending with the cell  $b + n + 4$ .

Operator  $C_2^j$ :

$$\begin{array}{ll} a + 2) d + 1^{**} 0000 c + 1 F \\ a + 3) b & 0000 c + 2 F. \end{array}$$

The addresses which are modified with respect to the parameter  $j$  will be denoted by a double asterisk.

Operator  $A_3^1$ :

/447

$$\begin{array}{l} a + 4) c + 2 c + 1 c + 2 M \\ a + 5) b + 1^* c + 2 c + 2 A. \end{array}$$

The address modification with respect to the parameter  $i$  will be denoted by a single asterisk.

After the instruction  $a + 5$  is executed, the quantity  $b_1 = b_{1-1} x_1 + a_1$  remains in the cell  $c + 2$ .

Operator  $F_4(i)$ :

$$a + 6) a + 5 e + 1 a + 5 A.$$

The auxiliary number 1 (I), by means of which the first address of the instruction  $a + 5$  is increased by a one, is loaded into the cell  $e + 1$ .

Operator  $P_5$ :

$a + 7) e + 2 e + 3 e + 25$   
 $a + 10) e + 2 \text{ 0000 0000 } C$   
 $a + 11) a + 12 a + 4 \text{ 0000 } CJ.$

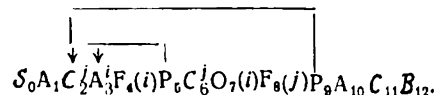
The cells  $e + 2$  and  $e + 3$  contain the auxiliary numbers  $(e + 2) = 4$  and  $(e + 3) = 1$ . With the aid of these numbers, a counter of loop repetitions is formed. Our loop should be repeated four times. After each completion of the loop, in accordance with the instruction  $a + 7$ , a one is subtracted from the tetrad written in the cell  $e + 2$ . The result of the subtraction is also written into the cell  $e + 2$ . In accordance with the instruction  $a + 10$ , the content of the cell  $e + 2$  becomes zero. If the cell  $e + 2$  contains a nonzero number, the comparator will generate the signal  $\omega = 1$  so that the succeeding instruction  $a + 11$  will transfer control to the initial instruction  $a + 4$  of the loop. After the loop has been repeated four times, cell  $e + 2$  will contain a zero, so that, after the instruction  $a + 10$  is executed, the signal  $\omega = 0$  will be produced. In accordance with this signal, control will be transferred to the next operator,  $C_6^j$ .

Operator  $C_6^j$ :

$a + 12) c + 2 \text{ 0000 } d + 1 ** F.$

The quantity  $y_j$  is loaded into a reply cell.

Let us now assemble the rest of the program. The complete logic diagram of the entire program has the form



In this diagram  $O_7(i)$  is the restoration operator. This operator restores the instruction  $a + 5$  of the operator  $A_3^j$ , which is address-modified with respect to the parameter  $i$ . The operator  $F_8(j)$  address-modifies the carry operators  $C_2^j$  and  $C_6^j$  with respect to the parameter  $j$ . Here,  $P_9$  is the logic operator which controls the external loop and ensures a sequential determination of the values of the function  $y$  for all values of the argument  $x = x_1, x_2, \dots, x_j, \dots, x_n$ ;  $A_{10}$  is the operator for converting the obtained data to the binary-coded decimal system of notation;  $C_{11}$  is the carry operator for transferring the results onto punch cards;  $B_{12}$  is the breakpoint operator.

We then assemble the second half of the program.

Operator  $O_7(i)$ :

$a + 13) e + 4 \text{ 0000 } a + 5 F.$

The restoration instruction  $a + 5$  is prestored in cell  $e + 4$ , in its original form

$$(e + 4) = b + 1 \quad c + 2 \quad c + 2A.$$

Operator  $F_8(j)$ :

/448

$$a + 14) \quad a + 2 \quad e + 1 \quad a + 2 \quad SA$$

$$a + 15) \quad a + 12 \quad e + 5 \quad a + 12 \quad SA.$$

The instruction  $a + 14$  address-modifies the instruction  $a + 2$  by means of the auxiliary number  $(e + 1) = 1$  (I). In accordance with the instruction  $a + 15$ , the third address of the instruction  $a + 12$  is modified, for which the auxiliary number becomes  $(e + 5) = 1$  (III).

Operator  $P_9$ :

$$a + 16) \quad a + 2 \quad e + 6 \quad 0000 \quad C$$

$$a + 17) \quad a + 20 \quad a + 2 \quad 0000 \quad CJ.$$

.The auxiliary number

$$d + 1 + n \quad 0000 \quad c + 1 \quad F.$$

is prestored in the cell  $e + 6$ .

This will coincide with the content of the cell  $a + 2$  after the instruction  $a + 2$  has been  $n$  times address-modified with respect to the parameter  $j$ . Any non-coincidence between the numbers  $(a + 2)$  and  $(e + 6)$  that is detected during their comparison will generate the signal  $\omega = 1$  which will cause the instruction  $a + 17$  to transfer control to the instruction  $a + 2$ , and the cycle will be repeated. After the external cycle has been repeated  $n$  times (and all the values of the sought function, from  $y_1$  to  $y_n$ , are computed), identical sets of digits will be present in the cells  $e + 6$  and  $a + 2$ , thus generating the signal  $\omega = 0$  and causing transfer of control to the next operator  $A_{10}$ .

Operator  $A_{10}$ :

$$a + 20) \quad d + 1 \quad n \quad d + 1 \quad 2 \rightarrow 10.$$

Operator  $C_{11}$ :

$$a + 21) \quad d + 1 \quad n \quad 0000 \quad Extr.$$

Operator  $B_{12}$ :

$$a + 22) \quad 0000 \quad 0000 \quad 0000 \quad K.$$

In the above example, the address modification operator  $F_7(i)$  may be used in place of the operator  $O_7(i)$ . The other operators of the logic diagram of the program remain unchanged.

To ensure proper work of the operator  $F_7(i)$ , the auxiliary number 3(I)

must be loaded into the cell  $a + 4$ .

Operator  $F_7(i)$ :

$a+13) a+5 e+4 a+5 SS.$

After this instruction is executed, the cell  $a + 5$  will contain, in place of the instruction

$b+4 c+2 c+2 A$

the original instruction

$b+1 c+2 c+2A.$

## Section 67. Logic Operators for Loop Control

Each loop includes a logic operator P controlling performance of the loop. Depending on the nature of the problem being solved, different methods of loop control may be used, as is evident from the above examples of cyclic program.

In this Section, we will discuss certain methods of loop control used in practical application.

The following notation will be adopted for their description:

/449

Loop segment preceding the Operator P	{	$a) \dots\dots\dots$
		$\dots\dots\dots$
		$\dots\dots\dots$
		$a + k) \dots\dots\dots$
Operator P	{	$a + k + 1) \dots\dots\dots$
		$a + k + 2) \dots\dots\dots$
		$\dots\dots\dots$

Here,  $a$  = instruction for starting the loop repetition,  
 $a + k$  = instruction transferring control to the operator P,  
 $a + k + 1$  = first instructions of the loop-controlling operator P.

Loop control by simple counter. Two variants of a loop counter exist.

The first variant contains two cells, one storing the number  $n$  which is the number of repetitions of the loop, and the other storing the number 1.

Suppose the number  $n$  is written into the cell  $c + 1$  and the number 1, in the cell  $c + 2$ . Then the logic operator P has the form:

$a + k + 1) c + 1 c + 2 c + 1 S$   
 $a + k + 2) c + 1 0000 0000 C$   
 $a + k + 3) a + k + 4 a 0000 CJ.$

For each loop repetition, in accordance with the first instruction of the operator, a one is subtracted from the number  $n$  and the result of this subtraction is written into the cell  $c + 1$ . According to the second instruction ( $a + k + 2$ ), the content of the cell  $c + 1$  is compared with zero. If  $(c + 1) \neq 0$ , this comparison results in generating the signal  $\omega = 1$ . According to the third instruction ( $a + k + 3$ ), control is transferred to the initial instruction  $a$  of the loop. After the loop has been repeated  $n$  times, the cell  $c + 1$  will contain a zero. Then, execution of the instruction  $a + k + 2$  will generate the signal  $\omega = 0$  and control will be transferred to the instruction  $a + k + 4$ , subsequent to the operator  $P$ .

The other variant of the counter contains three cells which store the number  $n$  (the number of loop repetitions), the number zero, and the number one, respectively.

Let us assume that these numbers are loaded into the following locations prior to operation of the counter:

$c + 1$	$c + 2$	$c + 3$
$n$	0	1

Then, the operator  $P$  will have the form:

/450

$a + k + 1) \ c + 2 \quad c + 3 \ c + 2 \ 5A$   
 $a + k + 2) \ c + 1 \quad c + 2 \ 0000 \ C$   
 $a + k + 3) \ a + k + 4 \ a \quad 0000 \ C \ J$

In accordance with the instruction  $a + k + 1$ , a one, written into the cell  $c + 3$ , is added to the content of the cell  $c + 2$ . The result of this addition is written into cell  $c + 2$ . After this, in accordance with the instruction  $a + k + 2$ , the numbers  $(c + 1)$  and  $(c + 2)$  are compared. If they do not coincide, control is transferred to the initial instruction  $a$  of the loop. After the loop has been carried out the first time, the number 1 will be written in the cell  $c + 2$ ; after the second time, the number 2, and so on. If the loop is repeated  $n$  times, the number  $n$  will be recorded in the cell  $c + 2$  and, after comparison, control will be transferred to the next instruction following the loop, namely, to  $a + k + 4$ .

Sometimes a modified second variant of the counter is used. In this case, the following numbers are written into the cells of the counter:

$c + 1$	$c + 2$	$c + 3$
$n \ (I)$	0	1 $(I)$

In this variant, the operator P has the form

$$\begin{array}{lll} a+k+1) & c+2 & c+3c+2 \text{ SA} \\ a+k+2) & c+1 & c+2 \text{ 0000 C} \\ a+k+3) & a+k+4a & \text{0000 CJ} . \end{array}$$

This last counter variant is convenient in cases where some address-modification constant is used in the programming. Then the cell  $c+3$  serves both to store this constant and to form the counter.

Loop control by a counter with variable instruction. This method is used to control address modification in loops. It has the advantage of requiring only one cell, into which the address-modified loop instruction in its final form is written, before constructing the counter. The loop-control operator compares the address-modified instruction with its final form and, if the compared numbers do not coincide, the operator transfers control to the initial instruction of the loop whereas, if they coincide, it transfers control to the next program instruction.

In this method of control, the entire loop, together with the operator P, may have the following general form:

/451

$$\begin{array}{l} a) \dots\dots\dots \\ a+i) \quad b+1^* \quad c+3b+1^*M \\ \dots\dots\dots \\ a+k) \dots\dots\dots \\ P \left\{ \begin{array}{lll} a+k+1) & a+i & e+1 \text{ 0000 C} \\ a+k+2) & a+k+3a & \text{0000 CJ} . \end{array} \right. \end{array}$$

The loop is repeated  $n$  times, with the address-modification operator standing next to the operator P. The address-modifiable instruction  $a+i$  of the loop is written into the cell  $e+1$  in the form which it will assume after the loop has been repeated  $n$  times, i.e.,

$$(e+1) = b + n \quad c + 3b + nM.$$

Loop control with repetition to satisfaction of an inequality. Let us assume that the solution of a problem must be continued until satisfaction of the condition

$$z - \epsilon \geq 0,$$

where  $z$  is a quantity determined with an accuracy regularly increasing after each loop and  $\epsilon$  is a preset number.

Assume that  $\epsilon$  is written into the cell  $b+1$  while the quantity  $z$  is loaded into the cell  $c+1$  after each execution of a loop.

Loop-control operator P:

$$\begin{array}{ll} a+k+1) c+1 & b+1 \text{ 0000 } S \\ a+k+2) a+k+3 a & \text{0000 } CJ. \end{array}$$

In accordance with the instruction  $a + k + 1$ , the difference  $\Delta = z - e$  is determined. If  $\Delta < 0$ , control is transferred to the initial instruction  $a$  of the loop since, in that case, the signal  $\omega = 1$ . As soon as the difference  $\Delta \geq 0$  is obtained, the signal  $\omega = 0$  is generated and control is transferred to the next instruction  $a + k + 3$  of the program.

In some cases, e.g., when solving problems by the iterative method, the computation must be continued until the condition

$|z| < \varepsilon,$

is satisfied, always specifying  $\epsilon > 0$ . This condition is satisfied when the inequality

$$|z| - |e| < 0$$

is satisfied.

Then the loop-control operator will contain the instructions:

$$\begin{array}{llll} a+k+1) & c+1 & b+1 & 0000 \text{ } S_1 \\ a+k+2) & a & a+k+3 & 0000 \text{ } CJ \end{array}$$

Loop control by self-erasing unconditional instructions. The loop-control operator P should contain as many self-erasing unconditional instructions as 452 the number of required loop repetitions. For example, take a loop of the form:

$$\text{Operator P} \begin{cases} a+k+1) \ a \ a \ a+k+1 \ C J \\ a+k+2) \ a \ a \ a+k+2 \ C J. \end{cases}$$

After first execution of the loop, the instruction  $a + k + 1$  will cause a repetition of the loop while at the same time writing a zero into the cell  $a + k + 1$ , i.e., will erase itself. Upon the second repetition of the loop the instruction  $a + k$  will be followed by execution of the instruction  $a + k + 2$ , since the cell  $a + k + 1$  will be filled with zeros. The instruction  $a + k + 2$  will again transfer control to the initial instruction  $a$  of the cycle while simultaneously loading the cell  $a + k + 2$  with zeros. Thus, the loop will be repeated for a third time, followed again by the next instruction  $a + k + 3$  of



the program.

This method of control is cumbersome if the number of loop repetitions is large, since the operator P must then contain a considerable number of instructions.

Loop control with two repetitions. Some loops must be repeated only twice. Any of the above-discussed logic operators may be used for the control of loops of this kind. However simpler methods of control are in existence, which will be given below.

Loop control by a "flipping" one. The number +1 is written into the cell  $a + 1$  and the number -1, into the cell  $c + 2$ . The logic operator P, controlling the loop, has the following form:

$$\begin{array}{l} a) \dots\dots\dots \\ a + k) \dots\dots\dots \end{array}$$

$$\text{Operator P} \left\{ \begin{array}{l} a + k + 1) \ c + 1 \ c + 2 \quad c + 1 \ M \\ a + k + 2) \ c + 1 \ c + 2 \quad 0000 \ C \\ a + k + 3) \ a \quad a + k + 4 \ 0000 \ C J. \end{array} \right.$$

After first execution of the loop, control is transferred to the instruction  $a + k + 1$  of the loop-control operator P. In accordance with this instruction, the numbers stored in the cells  $c + 1$ ,  $c + 2$  are multiplied out and the result of the multiplication  $(+1)(-1) = -1$  is written into the cell  $c + 1$ . After this, according to the instruction  $a + k + 2$ , the numbers  $(c + 1) = -1$  and  $(c + 2) = -1$  are compared. Since they are identical, the signal  $\omega = 0$  is generated and the next instruction  $a + k + 3$  transfers control to the initial instruction  $a$  of the loop. After the loop has been repeated, following execution of the instruction  $a + k + 1$ , the number  $(-1)(-1) = +1$  will be written into the cell  $c + 1$ . Then, in executing the instruction  $a + k + 2$ , the numbers +1 and -1 are compared as a result of which the signal  $\omega = 1$  is generated; according to the instruction  $a + k + 3$ , control is then transferred to the next instruction  $a + k + 4$  of the program.

/453

Loop control by a "flickering" number. A nonzero number  $\lambda$  will be prepared in the cell  $c + 1$ , while a zero is written into the cell  $c + 2$ . The logic operator P in this case will contain only two instructions:

$$\begin{array}{l} a + k + 1) \ c + 1 \quad c + 2 \ c + 2 \ C \\ a + k + 2) \ a + k + 3 \ a \quad 0000 \ C J. \end{array}$$

During the first execution of the loop, on the instruction  $a + k + 1$  the number  $\lambda$  is compared with zero and the result of the comparison, equal to  $\lambda$  itself, is written into the cell  $c + 2$ . Since  $\lambda \neq 0$ , the comparison generates the signal  $\omega = 1$  which causes the next instruction  $a + k + 2$  to transfer control to the initial instruction  $a$  of the loop. On repetition of the loop, the instruction  $a + k + 1$  causes the number  $\lambda$ , stored in the cell  $c + 1$ , to be com-

pared with the same number  $\lambda$  written into the cell  $c + 2$ . The result of the comparison - the number zero - will be loaded into the cell  $c + 2$ . At the same time, the signal  $\omega = 0$  will be generated and the instruction  $a + k + 2$  will transfer control to the next instruction  $a + k + 3$  of the program.

Both above control methods have the advantage of restoring the original form of the operator P following completion of the loop. Therefore, it is highly convenient to use these methods in the solution of problems requiring frequent use of a loop with two repetitions. This can be demonstrated on the following example.

Example. Let us assemble a program to determine the value Table of the function  $y = \sin(\sin x)$  for the values of the argument  $x = x_1, x_2, \dots, x_j, \dots, x_n$ .

The source data are allocated as follows:

$b + 1$	$b + 2$	$\dots$	$b + j$	$\dots$	$b + n$
$x_1$	$x_2$	$\dots$	$x_j$	$\dots$	$x_n$

The same locations are used for writing the results.

The logic diagram of the program reads

$$S_0 A_1 C_2^j A_3 P_4 C_5^j F_6(j) P_7 A_8 C_9 B_{10}$$

Here,

- $S_0$  = program input operator;
- $A_1$  = operator for conversion of source data to the binary system;
- $C_2^j$  = carry operator for feeding the new values of the arguments  $x_j$  to the arithmetic operator  $A_3$  determining the quantity  $y_j$ ;
- $P_4$  = loop-control operator;
- $C_5^j$  = operator transferring the quantity  $y_j$  to the reply cell;
- $F_6(j)$  = operator for address modification of the operators  $C_2^j$  and  $C_5^j$  with respect to the parameter  $j$ ;
- $P_7$  = logic operator for finding the function  $y$  for all values of the argument  $x$ ;
- $A_8$  = operator for converting the results to the binary-coded decimal /4<sup>5</sup> system;
- $C_9$  = operator for transferring the results onto punch cards;
- $B_{10}$  = breakpoint operator.

The operator  $P_4$  uses a "flickering" number, for which purpose the number  $\lambda$  is written into the cell  $c + 1$  and the number zero, into the cell  $c + 2$ .

Let us assemble the program for the segment from  $C_2^j$  to  $P_7$ :

Operator  $C_2^1$   $a + 1$   $b + 1^*$  0000  $c + 3$   $F$

•  $A_1$   $a + 2$   $c + 3$  0000  $c + 3$   $\sin x$

•  $P_4 \begin{cases} a + 3 \\ a + 4 \end{cases} \begin{cases} c + 1 \\ a + 5 \end{cases} \begin{cases} c + 2 \\ a + 2 \end{cases} \begin{cases} c + 2 \\ 0000 \end{cases} \begin{cases} C \\ CJ \end{cases}$

•  $C_3^1$   $a + 5$   $c + 3$  0000  $b + 1^*$   $F$

•  $F_6(j) \begin{cases} a + 6 \\ a + 7 \end{cases} \begin{cases} a + 1 \\ a + 5 \end{cases} \begin{cases} e + 1 \\ e + 2 \end{cases} \begin{cases} a + 1 \\ a + 5 \end{cases} \begin{cases} SA \\ SA \end{cases}$

•  $P_7 \begin{cases} a + 10 \\ a + 11 \end{cases} \begin{cases} a + 1 \\ a + 12 \end{cases} \begin{cases} e + 3 \\ a + 1 \end{cases} \begin{cases} 0000 \\ 0000 \end{cases} \begin{cases} C \\ CJ \end{cases}$

The following sets of digits are loaded into the cells  $e + 1$ ,  $e + 2$ ,  $e + 3$ :

$(e + 1) = 1(I),$   
 $(e + 2) = 1(III),$   
 $(e + 3) = b + n$  0000  $c + 3 F$ .

## Section 68. Features of Programming Based on the Use of Functional Tables

Calculations often involve dealing with various functions that must be determined in the process of problem solution. If these functions are analytically assigned, they are determined according to special subprograms or subroutines incorporated in the main program. At the required instant, the computer jumps to a subroutine of this kind and, after the function in question has been calculated, resumes executing the main program.

As pointed out previously, standard subroutines for the most frequently encountered functions ( $\sin x$ ,  $\cos x$ ,  $\sqrt{x}$ ,  $\log x$ , etc.) are assembled in advance for many digital computers and used on special instructions.

There exist, however, many functions which either cannot be expressed analytically or have no known analytic expression. Functions of this kind are assigned in the form of plots or Tables. Since a plotted function can always be used to compile a Table of its values, all functions of this kind can be considered tabular.

Programming based on table look-up has certain special features which will be the subject of this Section.

Prior to the calculations, a table of values of a function is loaded into the computer memory. If the Table is small, it is written into the immediate-access memory (internal memory unit). If it is large, it is recorded on magnetic tape.

When a table of values of a function is stored in the computer memory, /455 the main program is supplemented by a subroutine for selecting the required value of the function, taking into account interpolation between the tabulated values. If the functional Table is recorded on magnetic tape then, in addition

to these operations, provision is made for preselection of the required magnetic-tape zone and for copying the part of the Table in question, which requires additional time. Therefore recording a Table on magnetic tape is expedient in cases where access to the Table is rarely needed during solution of the problem.

If the functional Table cannot be completely accommodated in a given memory, it is stored in "rolled up" form in the memory, i.e., is contracted. Such convolution of Tables is a frequent measure since it will permit considerable savings in memory bulk.

The computation of the required value of a tabular function according to a "convolute" Table is a fairly complex problem which is easier to solve when the current values of the arguments are in agreement with the tabular values. However, this is a rare occurrence. Usually, the value of the argument does not coincide with the tabular value so that interpolation must be used, i.e., the function in the interval between two of its tabular values must be calculated.

In the general case, the function  $f(x)$  over a given interval is replaced by a polynomial of the  $n^{\text{th}}$  degree:

$$a_0 + a_1x + a_2x^2 + \dots + a_nx^n = P_n(x).$$

The degree and coefficients of the polynomial  $P_n(x)$  are so selected that, within the interpolation interval, the sought value of the function  $f(x)$  would approximately equal the value of  $P_n(x)$  with a permissible error of  $\epsilon$ . In other words, the condition

$$|f(x) - P_n(x)| \leq \epsilon.$$

must be satisfied. The polynomial  $P_n(x)$ , satisfying this condition, is known as an approximate polynomial.

In the interpolation intervals the entire tabulated function is represented by means of polynomial approximation. The selection of polynomials of this type is a fairly complex and laborious problem, usually handled by expert mathematician programmers.

Along with the tabular values of the function, the coefficients of approximate polynomials are stored in the computer memory. The subroutine for computing the tabular function  $f(x)$  permits finding the interval over which this function is assigned and computing the polynomial  $P_n(x)$  according to its coefficients stored in the memory.

In the most elementary case, the approximate polynomial  $P_n(x)$  is a polynomial of the first degree, which corresponds to a linear interpolation. As a specific example, we will discuss the procedure for assembling the subroutine /456 for computation of a tabular function. For better illustration, we will simplify the subroutine by using the method of linear interpolation. Usually, sub-

routines of this kind are more intricate because of the presence of an operator for computing the approximate polynomial.

Example. Given: 1) Table of values of the function  $y$  compiled with a constant step  $h$  of the argument  $x$ ;

2) Initial value of the argument:  $x_0$ .

Let us assemble the program for finding the function  $y$  according to the current value of the argument  $x$ , using the linear interpolation method.

When computing  $y$  according to the value of the argument  $x$ , the computer operates as follows: First, the tabular values  $x_i$  and  $x_{i+1}$ , between which the argument  $x$  is located, are derived. For this purpose, the difference between the value of  $x$  and the tabular values  $x_i$  of the argument, beginning with the initial value, is successively determined. If the difference  $x - x_i$  is positive, the next tabular value  $x_{i+1} = x_i + h$  is found and the difference is again determined. If the difference  $x - x_i$  is negative, this means that the current value of the argument  $x$  is located between the tabular values  $x_i$  and  $x_{i+1}$ . These tabular values correspond to the values  $y_i$  and  $y_{i+1}$  of the function. After this, the sought value of  $y$  is found by linear interpolation according to the formula

$$y = y_i + (x - x_i) \frac{y_{i+1} - y_i}{h}. \quad (54)$$

The source data for the problem, including the specified value of the argument, will be allocated as follows:

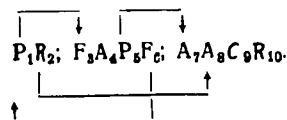
Cell No.	$b$	$b+1$	$b+2$	$\dots$	$b+l$	$\dots$	$b+n$	$b+n+1$	$b+n+2$	$b+n+3$
Cell content	$y_0$	$y_1$	$y_2$	$\dots$	$y_l$	$\dots$	$y_n$	$x_0$	$h$	$x$

We then load the cells  $e + 1$  and  $e + 2$  with the auxiliary numbers

$$\begin{aligned} (e + 1) &= 1(I), \\ (e + 2) &= 1(II). \end{aligned}$$

These numbers will be used for address modification. The working cells are denoted by  $c + 1$ ,  $c + 2$ ,  $c + 3$ ,  $\dots$  and the reply cell, by  $d + 1$ .

Assume that the source data are prestored in the computer. Then, the logic diagram of the program will be



In this diagram:

- $P_1$  = operator for comparing the specified value of the argument with the tabular values  $x_i$ ;  
 $R_2$  = operator for transferring  $y$  to the reply cell;  
 $F_3$  = operator for modifying the address of the operator  $R_2$ ;  
 $A_4$  = operator for determining the next tabular value  $x_{i+1} = x_i + h$ ;  
 $P_5$  = operator for determining the position of the assigned argument  $x$  among the tabular values  $x_i$ ;  
 $F_6$  = operator for modifying the address of the operator  $A_7$ ; /457  
 $A_7$  = interpolation operator, performing calculations in accordance with eq.(54);  
 $A_8$  = operator for converting the results to the binary-coded decimal system;  
 $C_9$  = operator for transferring the results to print-out;  
 $B_{10}$  = operator for halting the machine.

Program:

$P_1$	$\begin{cases} a+1) & b+n+1 & b+n+3 \\ a+2) & a+3 & a+5 \end{cases}$	$\begin{matrix} 0000 \\ 0000 \end{matrix}$	$\begin{matrix} C \\ CJ \end{matrix}$	$\begin{matrix} - \text{Compare } x \text{ with} \\ \text{tabular } x_i \end{matrix}$
$R_2$	$\begin{cases} a+3) & b^* & 0000 \\ a+4) & a+22 & a+22 \end{cases}$	$\begin{matrix} 0000 \\ 0000 \end{matrix}$	$\begin{matrix} F \\ CJ \end{matrix}$	$\begin{matrix} - \text{Transfer } y \text{ to reply} \\ \text{cell if } x = x_i \end{matrix}$
$F_3$	$a+5) & a+3 & e+1 & a+3$		$SA$	- Modify address of $R_2$
$A_4$	$a+6) & b+n+1 & b+n+2 & b+n+1$		$A$	- Obtain $x_{i+1} = x_i + h$
$P_5$	$\begin{cases} a+7) & b+n+3 & b+n+1 \\ a+10) & a+11 & a+14 \\ a+11) & a+14 & e+2 \end{cases}$	$\begin{matrix} 0000 \\ 0000 \\ 0000 \end{matrix}$	$\begin{matrix} S \\ CJ \\ SA \end{matrix}$	$\begin{matrix} - \text{Test for sign of the} \\ \text{difference } x - x_i \\ - \text{Modify address of the} \\ \text{instruction } a+14 \end{matrix}$
$F_6$	$\begin{cases} a+12) & a+21 & e+1 & a+21 \\ a+13) & a+1 & a+1 & 0000 \end{cases}$		$\begin{matrix} SA \\ CJ \end{matrix}$	$\begin{matrix} - \text{Modify address of the} \\ \text{instruction } a+21 \\ - \text{Return to beginning} \\ \text{of program} \end{matrix}$
	$\begin{cases} a+14) & b+1^* & b^* & c+1 \\ a+15) & b+n+1 & b+n+2 & c+2 \\ a+16) & b+n+3 & c+2 & c+2 \end{cases}$		$\begin{matrix} S \\ S \\ S \end{matrix}$	$\begin{matrix} - \text{Obtain the difference} \\ y_{i+1} - y_i \\ - \text{Compute } x_i + x_{i+1} - h \\ - \text{Compute difference} \end{matrix}$
$A_7$	$\begin{cases} a+17) & c+1 & c+2 & c+1 \\ a+20) & c+1 & b+n+2 & c+1 \end{cases}$		$\begin{matrix} M \\ D \end{matrix}$	$\begin{matrix} - \text{Compute the quantity} \\ (x-x_i)(y_{i+1}-y_i) \\ - \text{Compute the quantity} \\ (x-x_i)(y_{i+1}-y_i) \end{matrix}$
	$\begin{cases} a+21) & b^* & c+1 & d+1 \\ a+22) & d+1 & 0001 & d+1 \end{cases}$		$\begin{matrix} A \\ 2 \rightarrow 10 \end{matrix}$	$\begin{matrix} - \text{Compute the quantity } y \end{matrix}$
$C_9$	$a+23) & d+1 & 0001 & 0000$		$Extr.$	
$B_{10}$	$a+24) & 0000 & 0000 & 0000$		$K$	- Halt the machine

## Section 69. Checking Proper Execution of the Computational Process

The process of problem-solving by a digital computer consists of the following stages:

Assembling a working program.

Loading numerical source data into the machine.

Performing calculations in accordance with the prescribed program.

During all these stages, continuous monitoring of their proper execution is required, since the most insignificant error produced during one of these stages would lead to incorrect results. Thus, the distortion of a single digit in the address of an instruction or in the code of an operation alone will suffice to distort the result of the entire computation.

We will examine the methods of checking various stages of the computational process.

Checking the proper execution of the program starts with verifying the /458 writing of the program on blank forms. This check is handled either by the programmer himself or by someone else.

Then, the program is transferred onto punch cards (or punch tape). To provide for the possibility of a program check, the program is usually duplicated on two sets of punch cards independently by two persons who then compare the corresponding cards in both sets by means of a special device known as a verifier. Both sets of cards to be checked are inserted in the verifier where a special card gripper extracts identical cards from the sets which are then sensed by special contacts that close across the punches in the cards. If the cards prove to be nonidentical, they are removed from the verifier and replaced. This method of recording a program on punch cards is usually known as double-punch check.

In another method known as single-punch check, only one set of cards is punched. Then the stack of cards with the recorded program is inserted in an octal printer which reprints the program on paper tape. After this, the re-print of the program is compared with the program recorded on the blanks. Programs recorded on punch tape are verified in a similar manner.

Even if it is correctly recorded on the blanks and correctly transferred onto punch cards, a program may contain errors which were not originally detected during its assembly. In order to detect and correct these errors, so-called program adjustment is used in which the program is loaded into the computer memory while observing all necessary precautions for correct loading (see below). The computer then starts computing in accordance with this program and the obtained results are periodically verified in the course of these computations.

The correctness of a program can be verified by several methods.

One of these is to compare the results obtained by the computer with control data calculated in advance by mathematicians, relative to various computational stages. Determination of the control data is not a particularly laborious process, since the calculations are based on simplified source data taken equal to ones and zeros. At times, such data are given by experimental results in which case preliminary computations of this type are unnecessary.

Another method of adjustment is to verify the correctness of the program while the computer operates in the checkpoint regime. In this case, while writing the program on the blanks, check marks equal to unity are placed in the

instructions whose execution must be followed by halting the machine. After the machine is halted in accordance with the check mark, it transfers to the console the executed instruction, the number of the cell containing the next instruction, and the contents of the three cells indicated in the addresses of the /45 executed instruction. The checkpoints are so selected as to permit verifying the validity of the program logic (execution of necessary comparisons, transfer of control on conditional and unconditional instructions, etc.). At the same time during the halt, the results obtained by the computer can be compared with the control data obtained by the above methods.

The above-described methods of program adjustment are fundamental and are used frequently. There also exist other methods of program adjustment, but because of their great complexity they will not be considered here.

Verifying the proper loading of numerical source data into the machine.

The reliability of loading the source data and program is verified through a summation check of all numbers fed into the machine, by means of a special addition operation. In this case, all numbers and instructions fed to the machine are summed in succession. The resultant sums are known as check sums.

The numerical source data can be loaded into the computer by two methods: from two sets of punch cards (or punch tapes) or from a single set.

In the first method, the program is loaded successively from two identical sets of punch cards. First, the program is inserted from one set after which the first summation check is made. The first check sum is written into a special cell. This is followed by loading the program from the second set, which is written in place of the previously inserted program, and followed by a second summation check. After this, the two check sums are compared. If they agree, it is assumed that the program loaded from the second set is correct, and the machine starts execution of that program. If the check sums do not agree, the computer halts.

Loading a program from two sets of punch cards requires a great deal of time if the program is fairly large. Therefore, it is inconvenient to use this method when adjusting a program which must be repeatedly loaded into the computer.

In the second method of program loading, in which only one set of cards is used, proper loading is verified by comparing the two check sums obtained on double loading of one and the same program.

Checking the reliability of computations. The program itself makes a provision for verifying the proper operation of the computer in the process of problem-solving. The basic premise here is that the computer functions properly at the instant of startup and the program is adjusted.

Correct functioning of the computer is constantly checked by preventive maintenance during operation. In addition, special program checks known also as routine checks or test programs are run on the computer in order to further verify correct functioning. These tests are so laid out as to verify the /46C



proper functioning of each computer element.

Function checks of the computer by means of program tests are run prior to the computations or prior to adjustment of the program.

During problem-solving by the computer, the accuracy of the computations is usually verified by the method of double precision or double-triple arithmetic, or by the method of check relations.

Double-counting method. The machine solves the problem twice, comparing both results each time. If the quantity of the results is large, a summation check is performed after each solution, after which both check sums are compared. If the sums are in agreement, the solution is acknowledged to be correct. In the case of a large program, it is broken down into several parts, with each part executed twice. The results obtained after each counting are added up and the two check sums are compared. If the sums match, the machine proceeds to the next part of the program; if they do not, the machine halts.

Double-triple counting method. The part of program to be verified is calculated twice, with check sums being determined each time. Then these sums are compared. If they agree, the machine proceeds to the next part of the program. If the check sums do not agree, the calculations are repeated and a third check sum is determined. This sum is again compared with the preceding two check sums. If the third check sum agrees with one of the two preceding check sums, the result of the third counting is acknowledged to be correct and the machine proceeds to the execution of the next part of the program.

During computation, the integrity of the program must be checked, since any numerical data of the problem as well as the instructions themselves may become distorted while the computer is in operation.

If verification is performed by the method of double counting, distortion of the program during a miscalculation will lead to unnecessary halting of the machine and unproductive expenditure of time spent in locating the malfunction. In the method of double-triple counting, if a distortion occurs after the first counting, it may happen that the results of the second and third countings, although both incorrect, may coincide and thus be deemed correct.

Whatever program checkout is used, a check on program integrity is always mandatory. Usually this is done through a summation check of the program at prescribed stages of the computation, followed by comparison of the obtained check sum with the previously determined sum.

There exists another method of preserving program integrity, known as memory restoration. Prior to the computations, the program is written not only into the computer memory but also on magnetic tape. The program is so assembled that, after performance of certain computational stages, the check sum of the entire contents of the memory is determined. After this, the copy of the program on the magnetic tape is transferred to the computer memory and the counting is repeated, with another check sum of the entire contents of the memory. Then, the two check sums are compared. If they match, the computer

proceeds to the next stage of computation; if they do not, the computer halts.

Method of check relations. During solution of some types of problems it is possible to predetermine certain relations between the variables, relations that are not used in the counting process. Then, the accuracy of the computations can be verified by periodically checking whether these relations are satisfied. For example, during computation of a Table of sines and cosines, the correctness of the computational process can be checked by verifying whether the relation

$$\sin^2 x + \cos^2 x = 1.$$

is satisfied.

If the condition is satisfied, the computer results are correct.

This checkout method is more reliable than the one described earlier, since it simultaneously checks reliability of operation of the computer, integrity of the program, and accuracy of the computations. A program providing for verification of the computational process by the method of check relations is assembled for the solution of each individual problem.

# BIBLIOGRAPHY

1. Anisimov, B.V. and Chetverikov, V.N.: Fundamentals of the Theory and Designing of Digital Computers (Osnovy teorii i proektirovaniya tsifrovyykh vychislitel'nykh mashin). Mashgiz, 1962.
2. Batrakov, V.A. and Bogatyrev, V.I.: Electronic Digital Computers for Solving Information-Logical Problems (Elektronnyye tsifrovyye mashiny dlya resheniya informatsionno-logicheskikh zadach). Gosenergoizdat, 1961.
3. Belynskiy, V.V. and others: The Small-Scale Electronic Computer M-3 (Malogabaritnaya elektronnyaya vychislitel'naya mashina M-3). Izd. Akad. Nauk SSSR, 1957.
4. - The M-2 Rapid-Working Digital Computer (Bystrodeystvuyushchaya vychislitel'naya mashina M-2), edited by N.S.Brukh, GITTL, 1957.
5. - Computer Technology (Vychislitel'naya tekhnika). Collection of articles edited by S.A.Lebedev, Izd. Akad. Nauk SSSR, 1958.
6. Gitis, E.I.: Information Converters for Electronic Digital Computers (Preobrazovateli informatsii dlya elektronnykh tsifrovyykh vychislitel'nykh ustroystv). Gosenergoizdat, 1961.
7. Golovistikov, P.P. et al.: Arithmetic and Control Units of the BESM Computer (Arifmeticheskoye ustroystvo i ustroystvo upravleniya BESM). Fizmatgiz, 1960.
8. Gurvich, Ye.I. and Shchukin, L.B.: Ferrotransistor Elements and their Application in Automatic Digital Computers (Ferrotranzistornyye elementy i ikh primeneniye v tsifrovyykh avtomaticheskikh ustroystvakh). Gosenergoizdat, 1963.
9. Drozdov, Ye.A. and Pyatibratov, A.P.: Automatic Conversion and Coding of Information (Avtomaticheskoye preobrazovaniye i kodirovaniye informatsii). Sovetskoye Radio, 1964.
10. Zavolokina, Z.N.: Magnetic Elements in Digital Computers (Magnitnyye elementy v tsifrovyykh vychislitel'nykh ustroystvakh). Gosenergoizdat, 1958.
11. Zimin, V.A.: Electronic Digital Computers (Elektronnyye vychislitel'nyye mashiny). Mashgiz, 1962.
12. Itskhoki, Ya.S.: Pulse Units (Impul'snyye ustroystva). Sovetskoye Radio, 1959.
13. Kartsev, M.A.: Arithmetic Devices of Electronic Digital Computers (Arifmeticheskiye ustroystva elektronnykh tsifrovyykh mashin). Fizmatgiz, 1958.
14. Kitov, A.I. and Krinitskiy, N.A.: Electronic Digital Computers and Programming (Elektronnyye tsifrovyye mashiny i programmirovaniye). Fizmatgiz, 1959.
15. Laut, V.I. and Lyubovich, L.A.: Memory Unit with Cathode-Ray Tubes of the High-Speed Computer of the Academy of Sciences of the USSR (Zapominyushcheye ustroystvo na elektronno-luchevykh trubkakh bystrodeystvuyushchey schetnoy mashiny Akademii nauk SSSR). Izd. Akad. Nauk SSSR, 1957.
16. Lebedev, S.A. and Mel'nikov, V.M.: General Description of the BESM and Method of Executing the Operations (Obshcheye opisaniye BESM i metodika vypolneniya operatsiy). Fizmatgiz, 1959.

17. Martynov, Ye.M.: Noncontact Conversion Units (Beskontaktnyye pereklyuchayushchiye ustroystva). Gosenergoizdat, 1961.
18. Mayorov, F.V.: Electronic Digital Computers (Elements and Circuits) [Elektronnyye tsifrovyye vychislitel'nyye ustroystva (elementy i skhemy)]. Gosenergoizdat, 1957.
19. Mayorov, F.V.: Electronic Digital Computers and their Application (Elektronnyye vychislitel'nyye mashiny i ikh primeneniye). Voenizdat, 1959.
20. Pavlikov, A.A.: High-Speed Electronic Computer of the Academy of Sciences of the USSR; Magnetic Memory Unit (Bystrodeystvuyushchaya elektronnyaya schetnaya mashina Akademii nauk SSSR; Magnitnoye zapominayushcheye ustroystvo). Izd. Akad. Nauk SSSR, 1957.
21. Richards, R.K.: Arithmetic Operations in Digital Computers (Arifmeticheskiye operatsii na tsifrovyykh vychislitel'nykh mashinakh). IIL, 1957.
22. Richards, R.K.: Elements and Circuits of Digital Computers (Elementy i skhemy tsifrovyykh vychislitel'nykh mashin). IIL, 1961.
23. Smolenskiy, G.A. and Isupov, V.A.: Rochelle Salts and Similar Materials (Segnetoelektriki). Leningrad Publishing House of Scientific-Technical Literature, 1957.
24. - Technical Description of the "Ural" Automatic Digital Computer (Tekhnicheskoye opisaniye universal'noy avtomaticheskoy tsifrovoy vychislitel'noy mashiny "Ural"). GOSINTI, 1958.
25. Fel'dbaum, A.A.: Computers in Automatic Systems (Vychislitel'nyye ustroystva v avtomaticheskikh sistemakh). Fizmatgiz, 1959.

# TABLE OF CONTENTS

	Page
Authors' Preface .....	iii
Introduction .....	1
Chapter I. Arithmetic and Logical Principles of Electronic Digital Computers .....	6
Section 1. Positional Systems of Notation .....	6
Section 2. Arithmetic of Binary Numbers .....	9
Section 3. The Systems of Notation Used in Digital Computers .....	12
Section 4. Conversion of Numbers from the One Positional System of Notation into Another .....	14
Section 5. Representation of Numbers on Digital Computers ...	17
Section 6. Input of Numbers to the Memory Locations of a Digital Computer .....	21
Section 7. Representation of Negative Numbers .....	24
Section 8. Addition of Numbers on Computers .....	26
Section 9. Multiplication of Numbers on Computers .....	35
Section 10. Division of Numbers on Computers .....	37
Section 11. Elements of Mathematical Logic .....	41
Chapter II. General Principles of Digital Computer Design .....	52
Section 12. Types of Digital Computers .....	52
Section 13. Principal Units of an Automatic Digital Computer .....	54
Section 14. Structural Diagram of the Automatic Digital Computer .....	57
Section 15. Representation of Binary Numbers on Digital Computers .....	62
Chapter III. Elements and Units of Electronic Digital Computers .....	66
Section 16. Classification of Elements .....	66
Section 17. The Ferrite Core as a Bistable Element .....	68
Section 18. Basic Logic Elements using Ferrite-Diode Cells ...	77
Section 19. Ferrite-Transistor Cells .....	87
Section 20. Logic Circuits using Ferrite-Transistor Cells ....	95
Section 21. Basic Logic Elements from Electron Tubes and Semiconductor Devices. Circuits with Direct Connections .....	105
Section 22. Construction of Compound Logic Circuits .....	114
Section 23. Static Triggers .....	124
Section 24. Dynamic Flip-Flops .....	131

	Page
Section 25. Binary Counters .....	138
Section 26. Registers .....	149
Section 27. Selective Circuits (Decoders) .....	155
Section 28. Shifters .....	173
Chapter IV. Memory Units .....	178
Section 29. Types of Memory Units and their Main Characteristics .....	178
Section 30. Punch Cards and Punched Tapes .....	183
Section 31. Delay-Line Memory Units .....	189
Section 32. Memory Units Based on Cathode-Ray Tubes .....	193
Section 33. Magnetic-Drum and Magnetic-Tape Memory Units .....	203
Section 34. Matrix-Type Magnetic Working Memory Units .....	216
Section 35. Magnetic Working Memory Units with Half-Select Current Writing and Full-Current Reading .....	225
Section 36. Permanent Ferrite-Core Memory Unit .....	235
Section 37. Transfluxors .....	237
Section 38. Ferroelectric Memory Units .....	244
Chapter V. Arithmetic Units .....	248
Section 39. Arithmetic Units in General and their Principal Types .....	248
Section 40. One-Column Adders .....	249
Section 41. Coincidence Adders .....	261
Section 42. Decimal Adders .....	272
Section 43. Coincidence-Type Multiplication Units .....	276
Section 44. Accumulators .....	285
Section 45. Circuits for Operators of Multiplication and Division with Accumulators .....	292
Section 46. Coincidence-Type Arithmetic Units .....	299
Chapter VI. Input and Output Units .....	307
Section 47. Input and Output Devices in General .....	307
Section 48. Input Devices for Problem-Solving Computers .....	309
Section 49. Computer Output Equipment .....	319
Section 50. General Propositions on the Conversion of Con- tinuous Magnitudes into Discrete Quantities and of Discrete Quantities into Continuous Magnitudes .....	328
Section 51. "Voltage-To-Number" Converters .....	330
Section 52. "Shaft-To-Number" Converters .....	334
Section 53. "Number-To-Voltage" Converters .....	353
Section 54. "Number-To-Shaft" Converters .....	359
Chapter VII. The Control Unit .....	365
Section 55. General Characteristics of the Control Unit .....	365

	Page
Section 56. Control Units of Three-Address Computers .....	368
Section 57. The Control Units of One-Address Computers .....	375
Section 58. "Hard" Program Systems .....	382
Chapter VIII. The Elements of Programming .....	396
Section 59. The Structure of Instructions Used in Digital Computers .....	396
Section 60. Certain Facts on the General-Purpose Computer, Necessary for Programming .....	399
Section 61. The Instruction System .....	401
Section 62. Procedure for Preparation and Solution of Problems on the Machine .....	411
Section 63. Construction of Logic Diagrams of the Program ....	412
Section 64. Compilation of Programs in Conditional and Real Addresses .....	417
Section 65. Branching Programs .....	420
Section 66. Cyclic Programs .....	424
Section 67. Logic Operators for Loop Control .....	437
Section 68. Features of Programming Based on the Use of Functional Tables .....	443
Section 69. Checking Proper Execution of the Computational Process .....	446
Bibliography .....	451